Super-Resolution in Still Images and Videos via Deep Learning

Farzad Toutounchi

Submitted in partial fulfillment of the requirements of the Degree of Doctor of Philosophy

School of Electronic Engineering and Computer Science
Queen Mary University of London
United Kingdom

August 2019

Statement of Originality

I, Farzad Toutounchi, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Farzad Toutounchi

August 2019

Details of collaborations and publications:

All papers published while working on this thesis are listed at the end of Chapter 1. Any publication produced in collaboration with others is clearly mentioned.

Abstract

The evolution of multimedia systems and technology in the past decade has enabled production and delivery of visual content in high resolution, and the thirst for achieving higher definition pictures with more detailed visual characteristics continues. This brings attention to a critical computer vision task for spatial up-sampling of still images and videos called super-resolution. Recent advances in machine learning, and application of deep neural networks, have resulted in major improvements in various computer vision applications. Super-resolution is not an exception, and it is amongst the popular topics that have been affected significantly by the emergence of deep learning. Employing modern machine learning solutions has made it easier to perform super-resolution in both images and videos, and has allowed professionals from different fields to upgrade low resolution content to higher resolutions with visually appealing picture fidelity. In spite of that, there remain many challenges to overcome in adopting deep learning concepts for designing efficient super-resolution models.

In this thesis, the current trends in super-resolution, as well as the state of the art are presented. Moreover, several contributions for improving the performance of the deep learning-based super-resolution models are described in detail. The contributions include devising theoretical approaches, as well as proposing design choices that can lead to enhancing the existing art in super-resolution. In particular, an effective approach for training convolutional networks is proposed, that can result in optimized and quick training of complex models. In addition, specific deep learning architectures with novel elements are introduced that can provide reduction in the complexity of the existing solutions, and improve the super-resolution models to achieve better picture quality. Furthermore, application of super-resolution for handling compressed content, and its functionality as a compression tool are studied and investigated.

Acknowledgements

First and foremost, I have to thank my supervisor Ebroul Izqueirdo, who offered me a position in his group to pursue a PhD. My time at MMV group under his supervision has been a real journey with a great deal of learning. What remains for me after leaving MMV would be far more than the research I did and the papers I published. Doing a PhD was an opportunity to experience real team work, and a chance to acquire a variety of skills that will come in handy in any future endeavor.

I should also thank the COGNITUS consortium, and everybody in that project who I had a chance to work with. COGNITUS was the main source of funding for my research, and it was a great platform to collaborate with industrial partners, and do research and development work on some real world challenging tasks.

It goes without saying that I'm humbled and grateful to have worked along with all the current and previous talented MMV members. There are too many to acknowledge, and I can only say a big thank you to everyone. Every chat, every lunch, and every drink was a real treat at MMV, and having everybody around made the student life way easier.

I have to give a big shout out to all my friends all over the world. They are simply the best I can ask for, and I'm so lucky to have them. I watched every single one of them grow and excel in their personal and professional lives, and couldn't be more proud. They have always been a true source of inspiration and happiness in my life.

Last but not least, I have to thank my family, who are always so kind and loving. Although they all live so far away, they remain always in my heart, and I want to thank them over and over again for their relentless support and countless sacrifices in my entire life. I wouldn't be who I am without them. I would like to dedicate this thesis to them, although they would probably never read it.

Contents

List of Figures

List of Tables

List of Abbreviations

ALPN	**A**ccurate **L**ossless **P**ooling **N**etwork
AVC	**A**dvanced **V**ideo **C**oding
BD	**B**jøntegaard **D**elta
CNN	**C**onvolutional **N**eural **N**etwork
CR	**C**ompression **R**atio
CTU	**C**oding **T**ree **U**nit
CU	**C**oding **U**nit
FLPN	**F**ast **L**ossless **P**ooling **N**etwork
GAN	**G**enerative **A**dversarial **N**etwork
HD	**H**igh-**D**efinition
HEVC	**H**igh **E**fficiency **V**ideo **C**oding
MLP	**M**ulti **L**ayer **P**erceptron
MPC	**M**odified Proximity-based **C**ost
MSE	**M**ean **S**quared **E**rror
MV	**M**otion **V**ector
PSNR	**P**eak **S**ignal-to-**N**oise **R**atio
PU	**P**rediction **U**nit
QP	**Q**uantization **P**arameter
ReLU	**Re**ctified **L**inear **U**nit
RQT	**R**esidual **Q**uad-**T**ree
SGD	**S**tochastic **G**radient **D**escent
SR	**S**uper-**R**esolution
SSIM	**S**tructural **SIM**ilarity
TU	**T**ransform **U**nit
UET	**U**ltra **E**ye **T**racking
UHD	**U**ltra **H**igh-**D**efinition

Chapter 1

Introduction

1.1 Motivation

In recent years, audio-visual equipment has become pervasive, offering everyone accessibility to consume media services through a range of smart devices including mobile phones, tablets and large television sets. Among the highly-segmented market of new generation display devices, the 4K Ultra High-Definition (UHD) has attracted larger consumer awareness in the last few years according to industry analysts. This trend is expected to continue spiraling upwards as more households and other settings are infiltrated by display or video capturing devices with UHD capability and beyond[1].

The scientific innovation in the development of UHD is primarily driven by the market need to deliver high-quality viewing experience from broadcaster services. This initiative is suitably complemented by the efforts in international standardization[2].

The emergence of UHD has seen global adaptation across a wide range of market segments, including broadcasters and video streaming service providers. Across Europe alone, there are more than 20 television channels that offer UHD services, and there exist nearly 50 UHD channels worldwide[3], while the streaming service providers such as Netflix and Amazon have also started providing content in UHD format. However, the

[1] http://4k.com/news/4k-tv-market-to-reach-52-billion-in-revenues-by-2020-6655/

[2] https://www.dvb.org/standards/dvb-mpeg-uhd

[3] https://en.wikipedia.org/wiki/Ultra-high-definition_television#List_of_4K_television_channels

majority of the available content is still in non-UHD format, and a significant amount of content is still being captured and rendered in older standard formats. This raises the challenging issue of format conversion and content adaptation for UHD services, which requires tools for spatial re-sampling of video signals.

Spatial up-scaling has attracted a lot of attention in the broadcasting industry, while many other multimedia applications also require similar measures for not only adapting the spatial resolution of the images and videos to a target format, but also content restoration and quality enhancement of low quality and corrupted data. Archiving and footage restoration are amongst the active and important industries in need of spatial re-sampling, that revive significant amounts of historical and legacy material in both forms of still images and videos. Spatial up-sampling along with enhancement mechanisms play crucial roles when handling archive content, whether they are aimed to be used for research purposes, or exploited in artistic and creative environments.

Super-Resolution (SR) is a computer vision approach for performing high quality spatial up-sampling in still images and videos, capable of adapting low resolution content to a desired higher resolution setting, while retaining the visual consistency of the content, and possibly improving the picture fidelity. SR is an effective tool for spatial adaptation of the non-UHD material to UHD format, as well as up-scaling user-generated content, and restoring archive footage. It can provide a positive impact on the quality of service for broadcasting and entertainment applications, as well as improvements in automatic restoration work-flows for archiving purposes.

Application of SR, however, is not limited to the above-mentioned multimedia domains, and it can be adapted to deliver spatial up-sampling and content enhancement solutions in a wide range of applications including and not limited to enhancing security and closed-circuit television footage [41, 47, 132], improving remote sensing and satellite imaging [54], as well as astronomical observations [52, 74], and effective enlargement of medical imagery [25, 67, 141].

SR has been studied extensively in the past decades, and numerous approaches have been proposed to perform high quality image and video restoration effectively. Given the nature of the SR problem, which deals with information recovery and prediction of signal behaviors, machine learning has been a candidate solution to provide answers to the many challenging aspects of the task. In particular, deep learning approaches

have been extremely successful in developing SR solutions, as they have been in many other computer vision tasks. Introduction of Convolutional Neural Networks (CNN) and various generative architectures to the SR has revolutionized this field and created promising prospects for efficient and high quality spatial up-sampling of still images and videos. Application of deep learning in SR has been considered from different perspectives, and variety of approaches are introduced based on novel trainable CNN architectures.

Despite the significant progress in this field, SR remains an active research field, with many challenges yet to be overcome. The next section lists some of the key challenges and problems in the existing deep learning-based SR solutions, and discusses the need for conducting further research to achieve more effective systems for content restoration and spatial up-sampling.

1.2 Challenges

Deep learning models are trained machines that have various capabilities including classification, regression, and prediction. In some applications such as regression, a deep learning model can be represented as a mapping between two spaces. This is true in particular for SR task, in which a low resolution signal is mapped to a high resolution version of the same content. When studying tools derived from deep learning, there are always two questions to be answered: How accurate is the model? And, how complex is the model?

Accuracy of the model can typically be quantified by a score that measures the performance of the trained machine in predicting the correct outcome, when dealing with new data. In classic machine learning problems and classification algorithms, accuracy is computed by counting the true positives, whereas in regression problems the accuracy is usually characterized by computing the norm of the difference between the output and a ground truth. In case of SR, the accuracy is the visual quality of the picture which can be measured by comparison with a ground truth and quantified by metrics such as Peak Signal-to-Noise Ratio (PSNR) and other similarity indexes.

Complexity of the model characterizes the practicality of the trained machine in handling problems. Specifically, the computation time of the processes in an end-to-end model

implemented on a particular hardware can play a major role in applicability of the model for real world scenarios. In SR task, the complexity is typically measured by the amount of time needed to infer spatial up-sampling for a given signal, and the metric can be dependent on the source and target resolution, as well as the deployed hardware.

Both accuracy and complexity are topics that are being studied and investigated actively in deep learning-based SR, and major advances have been reported in the literature in providing high quality SR methods with low complexity for still image and video applications. Nonetheless, the research in this direction cannot stop, as the demand to have more accurate models with better restoration performance is increasing every day, and there are more applications and scenarios that can benefit from better SR models. With regard to complexity, having practical and usable models is of major importance. Given the correlation of the complexity with the desired target resolution, and the increasing interest in having content in UHD resolution and beyond, the research in complexity reduction of SR methods is becoming more intriguing.

Addressing both accuracy and complexity in one system comes with challenges, as improving one can often lead to deterioration of the other, and designing novel deep learning-based models, or enhancing the existing SR architectures requires careful attention to many details and subtleties that play crucial roles in the performance of the system. Recent improvements in SR models are largely thanks to complex models that have millions or tens of millions of parameters. Training and tuning the model parameters is also an extremely complicated and challenging stage, which requires both theoretical considerations, as well as optimized implementations. Hence another active research direction in deep learning would be to carry out investigations on improving the training process in SR models, which can lead to either better models in terms of accuracy, or easier and faster training cycles for complex models.

The domain, in which the SR model is going to be deployed, also plays an important role in the design and training stages. The training data and the tuning process can (and in some cases must) be adapted to the desired application, and that leads to more challenges such as lack of data, design deficiencies, and complications in training. This problem is magnified when applying SR in new and untested domains.

1.3 Contributions

In this thesis, the state-of-the-art approaches in still image and video SR are discussed with focus on deep learning-based solutions. The pioneering work, along with the key concepts applied in existing SR models are summarized. Moreover, several contributions by the author are introduced and described in details, that can enhance and improve the current SR approaches from different perspectives. The following are the main contributions presented in this thesis.

1. A novel cost function for training SR convolutional networks is introduced that leads to increased training efficiency and can significantly reduce the number of necessary back-propagations during the training.

2. A novel pooling layer is proposed that reduces the spatial size of the feature maps without losing information by performing a reshuffling mechanism that increases the depth of the tensors. Moreover,

   - A CNN architecture is designed for still image SR that takes advantage of the novel pooling layer, and can be adapted to any encoder-decoder architecture to reduce the complexity and computation time of inference, particularly for higher resolutions.
   - A CNN architecture is designed for still image SR that takes advantage of the novel pooling layer, and can be adapted to any encoder-decoder architecture to improve the quality of the picture and lead to more accurate reconstruction using self-similarities within the input signal.

3. A single image SR deep learning architecture is adapted for efficient video up-scaling from Full HD resolution to 4K UHD resolution. In that regard, a data preparation process is introduced for handling compression artifacts for training SR models adapted to compressed videos. Moreover, an SR-based compression pipeline is designed that can provide the same picture quality as state-of-the-art encoding approaches for UHD video compression.

1.4 Thesis Structure

This thesis consists of six chapters, with the first two describing introductory and preliminary topics in deep learning-based SR research. The main contributions of the author are presented in Chapters 3-5. The following is the thesis organization in more details.

Chapter 2 provides a literature review on still image and video SR with a focus on deep learning approaches, and summarizes the key concepts applied today in existing SR models.

Chapter 3 discusses designing a novel cost function for efficient and fast training of SR convolutional networks, that can promise significant reductions in training time of the well-known architectures. This chapter covers contribution 1.

Chapter 4 introduces the novel lossless pooling layer for CNNs, and provides details on designing SR deep learning-based architectures using the novel pooling layer for fast and accurate image up-scaling. This chapter covers contribution 2.

Chapter 5 proposes a modified still image SR model for performing high quality efficient SR on compressed videos from Full HD resolution to 4K UHD resolution, with considerations on application of SR in video coding. This chapter covers contribution 3.

Chapter 6 concludes the thesis with summary of the presented work, as well as providing ideas on prospects of the research in the SR field and possible future developments.

1.5 Published Work

The author has published several peer-reviewed papers in international conferences and journals during the course of his PhD research, some of which are directly related to the topics discussed in this thesis.

- **F. Toutounchi**, E. Izquierdo, "Efficient Training of Super-Resolution Convolutional Networks," *IEEE Transactions on Image Processing*, 2019. (Under review)
- M. Giannopoulos, G. Tsagkatakis, S. Blasi, **F. Toutounchi**, A. Mouchtaris, P. Tsakalides, M. Mrak, E. Izquierdo, "Convolutional Neural Networks for Video

Quality Assessment," *Elsevier Journal of Signal Processing: Image Communication*, 2019. (Under review)

- **F. Toutounchi**, E. Izquierdo, "Advanced Super-Resolution using Lossless Pooling Convolutional Networks," *Winter Conference on Applications of Computer Vision*, pp. 1562–1568, Jan 2019.

- **F. Toutounchi**, E. Izquierdo, "Enhancing Digital Zoom in Mobile Phone Cameras by Low Complexity Super-Resolution," *International Conference on Multimedia and Expo Workshops*, pp. 1–6, Jul 2018.

- **F. Toutounchi**, E. Izquierdo, "An Open Access User Generated Video Dataset from 2016 Edinburgh Festival," *Latin American Conference on Networked Electronic Media*, pp. 12–15, Nov 2017.

- **F. Toutounchi**, V. Guerra Ones, E. Izquierdo, "An Efficient Super-Resolution Approach based on Sparse Representation," *International Workshop on Multimedia Signal Processing*, pp. 1–6, Oct 2017.

Chapter 2

Background on State-of-the-Art Super-Resolution

This chapter presents an overview of Super-Resolution (SR) as a computer vision paradigm, with a focus on learning-based approaches, and in particular methods utilizing Convolutional Neural Networks (CNN) as their core engine. While describing some of the key concepts in application of machine learning in SR, the most recent research work in this field is mentioned and a comprehensive literature review on deep learning-based image and video restoration is presented.

2.1 Introduction

The main goal of spatial up-sampling is to enlarge still images and videos and create a spatially up-scaled version from low resolution content with satisfactory visual quality. Spatial up-sampling and content restoration have been studied actively in recent years, and a significant amount of research has been carried out in these areas. Although the advances in this field provide promising results, performing high quality restoration and up-scaling remains an active and challenging topic in computer vision.

Two general approaches are considered, when applying spatial up-scaling: Interpolation-based approaches, and SR-based approaches. Interpolation-based approaches are easy solutions to up-scaling problem, and address the issue by creating a larger version of

the image or video using pre-defined interpolation filtering operations such as nearest-neighbor, bi-linear, bi-cubic, etc. Such approaches are computationally efficient, but do not provide favorable picture quality, given the over-simplified characteristics of the filters and their tendency to generate smooth textures. However, they are the method of choice for many format conversion and content adaptation applications due to their low complexity nature.

The other category of approaches are the SR-based methods. SR is a more complicated approach in performing spatial up-scaling that results in generating pictures with higher visual quality. In other words, the distortions introduced by performing interpolation-based methods tend to be avoided in SR-based approaches, and the visual consistency of the picture is retained when up-scaled. SR-based approaches can be classified into two types [129] of: Reconstruction-based (model-based) methods, and example-based (learning-based) methods. SR problems can be formulated as the following:

$$\mathbf{Y} = SH\mathbf{X} + n, \tag{2.1}$$

where $\mathbf{Y}$ represents the low resolution image, $\mathbf{X}$ represents the high resolution image, and S and H represent down-sampling and blurring operations, respectively. In the above equation, n is additive Gaussian noise. Equation 2.1 represents the low-resolution image as a down-sampled and blurry version of the high-resolution image, with addition of some noise. SR approaches aim at solving the above equation for $\mathbf{X}$ when $\mathbf{Y}$ is given. It is, however, worth noting that the above equation is the most generic way of formulating SR, and is not representative of the advanced models and approaches (e.g. deep learning) that are being used today for handling SR problems.

In reconstruction-based approaches, the image formation process is simulated, and a model is designed to represent the reverse process of down-sampling and blurring operations. It is perhaps fair to say that interpolation-based up-scaling methods are a simplified version of reconstruction-based SR approaches. There have been various contributions in devising model-based single image and multi-frame SR models for high quality image up-sampling [14, 15, 34, 44, 59, 78, 83], however in recent years with significant improvements in application of machine learning it is no surprise that SR methods are also impacted tremendously by new learning- and example-based approaches.

In example-based approaches, the relation between the low and high resolution image segments are learned using a set of training data that is representative of different natural textures in variety of visual scenarios. While reconstruction-based approaches are proved to provide rather good results in terms of image quality, they can often be very inefficient in terms of needed computational power. Example-based approaches, on the other hand, often reconstruct the high-resolution image in a block-wise routine, hence all the operations are performed on blocks (patches) of pixels, resulting in complexity reduction in the SR process. Recently, example-based techniques have been improved further by introduction of state-of-the-art deep learning techniques, and the performance of the spatial up-sampling tools have been enhanced significantly both in terms of reconstructed picture quality and the efficiency of the approaches [30–32]. Deep learning-based approaches have enabled new and promising prospects for SR and its applications in different domains.

As this thesis is mainly focused on deep learning-based SR approaches, this chapter provides a short survey on the most significant contributions in application of CNNs in still image and video SR, while describing some of the key theoretical concepts introduced and applied in recent years. Starting with a summary of general example-based SR approaches in Section 2.2, the chapter continues with presenting the pioneering work in deep learning-based single image SR in Section 2.3, followed by introduction of prominent research work in multi-frame SR approaches based on deep learning in Section 2.4. Section 2.5 reviews the main metrics for evaluating the performance of the SR methods. The chapter concludes with Section 2.6, which is a concise summary of the presented approaches.

2.2 Conventional Learning-Based Super-Resolution

In the following, some of the key learning-based contributions prior to emergence of deep learning are discussed briefly, and the main algorithms for applying example-based concepts in SR for either still images or videos are outlined. As mentioned earlier, example- or learning-based approaches in SR can learn representations for visual features and learn a mapping between the low resolution and high resolution spaces. Categorically, learning-based approaches either exploit the internal similarities [35, 40, 127], also known as self-similarities, or learn from external paired low resolution and high resolution data

sets [24, 36, 58, 107, 111, 128, 129], in which case the approach would require a large amount of training data.

With regard to the learning-based approaches exploiting self-similarities, Freedman and Fattal [35] introduced an approach that follows a local self-similarity assumption on natural images and extracts patches from extremely localized regions in the input low resolution image. This leads to complexity reduction in performing nearest-patch search with minimal compromise on the picture quality. Yang et al. [127] provided an approach that combines the two fundamental learning-based approaches of learning from an external database, and learning from self-examples. Based on the existing in-place examples, a first-order approximation of the nonlinear mapping between low and high resolution pixel blocks is learned. The application of self-similarities were even extended further to more recent deep learning-based models in [48] and [102].

One of the interesting approaches in learning-based SR is based on the sparse coding of images [91]. Sparse coding is the process of representing observed signals with combination of only a small number of basis elements chosen from a set of candidates called a dictionary. In sparse coding all the processes are performed on blocks of pixels and the high-resolution image is reconstructed by integrating the high-resolution blocks. The correspondence between the low-resolution and high-resolution dictionaries is established through a set of assumptions. A low-resolution block is represented by combining a set of low-resolution bases and the same coefficients are applied for the high-resolution bases to be combined and reconstruct the high-resolution block. Yang et al. [129, 130] were the first to apply sparse representation to improve the approach presented in [24].

Yang et al. [128] improved their initial presented approach further by introduction of coupled dictionary training for still image SR. Moreover, Zeyde et al. [136] introduced further modifications to the sparse representation-based algorithm and training process that led to complexity reduction in the image SR approach. The sparse coding method for SR is not limited to still images, and it was extended to videos (multi-frame SR) by Song et al [104]. Dai et al. [28] and Kato et al. [62] made similar attempts for designing multi-frame SR models using sparse coding paradigm by improving the dictionary learning process, and motion compensation stages of the approach.

Toutounchi et al. [114] contributed in complexity reduction of the sparse representation-based SR methods for applications in high resolution videos. They introduced an approach that enables skipping the sparse representation process for the static portions of a video by predicting the high resolution blocks from previously up-scaled frames, exploiting the significant amount of correlation between the adjacent frames in a video. In addition, the approach utilizes adaptive in-loop filters [37, 90] for removing blocking and pixel-wise artifacts, replacing the overlapping blocks structure of SR.

In other learning-based approaches, Timofte et al. [111] proposed a fast SR method that uses sparse representation in combination with neighbor embedding method [24]. The nearest neighbors are computed using the correlation with the dictionary cells rather than the Euclidean distance. Moreover, the method takes advantage of the global collaborative coding that leads to significant speed-ups in the performance. Additionally the authors proposed application of anchored neighborhood regression. Overall the method provides favorable picture quality with reasonable complexity.

Timofte et al. [112] improved their anchored neighborhood regression method further by combining the concept with simple functions method [126]. While the anchored neighborhood regression learns sparse dictionaries and regressors anchored to the dictionary cells, simple functions method relies on clusters and corresponding learned functions. The combined method builds on the features and anchored regressors but instead of learning the regressors on the dictionary, it uses the full training material, similar to simple functions. The performance of the model improves over the baseline regression-based method both in terms of picture quality and computation complexity.

Application of random forests [13, 20, 26] have also been considered for performing single image SR. Schulter et al. [99] proposed a direct mapping model from low to high resolution image patches using random forests. The approach is closely related to the linear regression methods described earlier, and training is done by optimization of a novel and effective regularized objective. The inference of the proposed SR method is quite efficient by application of the random forests, and the picture quality is promising.

Salvador and Pérez-Pellitero [95] were also amongst the firsts to adopt random forests for SR applications. Their contribution was a bi-modal tree for clustering, which successfully

exploits the antipodal invariance[1] of the coarse-to-high resolution mapping of natural image patches and provides scalability to finer partitions of the underlying coarse patch space. The second and main contribution was a fast inference algorithm, which selects the most suitable mapping function within the tree ensemble for each patch by adopting a local naïve Bayes [73, 137] formulation.

More recently, Zhi-Song and Siu [142] proposed a cascaded random forest still image SR which screens sufficient simple features to train a robust and efficient model. As an extra measure, an auxiliary Gaussian mixture model layer is also added as a final refinement step to further boost up the performance. The proposed method demonstrates promising results, and the performance tends to be better when more features are selected for refinement.

This section provided a variety of contributions and methods that build upon example-based approaches for performing single image or multi-frame SR. It is no surprise that the advent of deep learning and CNNs have also impacted the SR field significantly. Consequently, the majority of the research carried out in this field in recent years has been relying on deep learning concepts. The next section provides a summary on the most significant contributions in SR based on deep learning, while describing the underlying theoretical concepts.

2.3 Deep Learning-Based Single Image Super-Resolution

CNNs have proved to be powerful tools for computer vision tasks that require intelligent prediction mechanisms. Although CNNs were already introduced in the past decades [71], it is only in recent years that they have gained popularity and their functionalities have been applied in various machine learning tasks. Several aspects support the explosive growth of deep learning in computer vision, amongst which application of powerful GPUs [66] can be named as a key factor. In addition, introduction of optimized architectures and efficient operations such as Rectified Linear Unit (ReLU) [88], as well as abundance of the training data thanks to the availability of large open-access data

[1]Two points on the surface of a sphere are antipodal, if they are diametrically opposite. Antipodal invariance refers to a feature in metrics and objective functions, where a pair of antipodal points are treated similarly regardless of their vector direction.

sets such as ImageNet [29] have made the training process of CNNs easier and more efficient.

CNNs consist of several operational layers that perform matrix operations on the input signal, and each layer maps its input to a new space, producing feature maps that can represent various features of the input signal. Amongst the various existing layers in a typical CNN, perhaps convolutional layers are the most famous ones that can each consist of n convolutional filters with equal kernel sizes being imposed on the input signal. In addition to the filtering operation a bias vector is added to the filters output obtained from the n available filters. Moreover, an activation function is applied to provide non-linearity. The activation function can be selected from a variety of choices including ReLU, Parametric ReLU [45], tanh, etc. The following formulates the behavior of a single convolutional layer with ReLU activation:

$$F(\mathbf{Y}) = \max(0, \mathbf{W} * \mathbf{Y} + \mathbf{B}) \tag{2.2}$$

where F represents the end-to-end behavior of the layer, and $\mathbf{Y}$ represents the input matrix (tensor). $\mathbf{W}$ and $\mathbf{B}$ are the filter kernels and the bias vector, respectively. If the input tensor has a shape of $W \times H \times C$, with W, H, and C representing the width, height, and number of channels for a sample input image, then the output signal will be a tensor with the shape $W \times H \times n$, given that enough padding is applied on the input signal to ensure the same spatial size for input and output. It is also worth noting that convolutional layers can have different stride parameters, that specify the positioning of the kernel on the input signal. Different stride values can lead to different output shapes, e.g. a stride of 1 ensures that the kernel is positioned on every pixel of the input signal, while a stride of 2 will skip every other pixel during the convolution operation. Figure 2.1 illustrates a visual presentation of a convolutional filter without the bias and ReLU activation function. The input image has the $8 \times 8 \times 1$ shape, and the filter has a 5×5 kernel. Full padding is applied to the input image to keep the output shape equal to the input shape.

By having multiple layers within a deep learning architecture, the model can be designed to generate a final output with the desired shape, and by solving an optimization problem for a particular objective function, the network parameters can be trained to provide a model that produces the envisaged output. In SR, the input and output signals are in

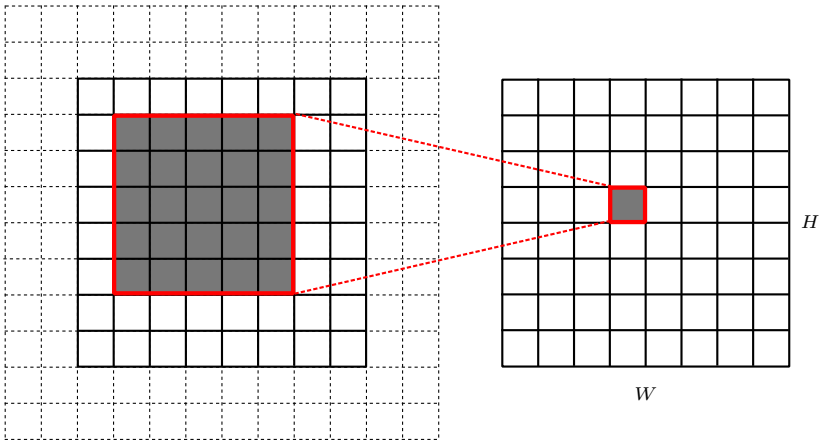


FIGURE 2.1: The visual presentation of convolutional filtering. The dashed lines represent the paddings applied to the input signal.

the same pixel space, hence a class of CNNs called encoder-decoder and auto-encoders (that are a special type of encoder-decoders) are used for performing SR, that can map a low resolution (or low quality) image to a high resolution (or high quality) image.

Deep learning and Multi-Layer Perceptron (MLP) were previously used for image restoration in several research contributions including for image de-noising by Burger et al. [22], and Schuler et al. [98]. However the first concrete CNN-based approach for SR that employed an end-to-end trainable deep learning architecture was proposed by Dong et al. [30, 31], that ignited the ever-growing application of CNNs in single image and multi-frame SR.

The SRCNN model proposed by Dong et al. [30, 31] is a flat auto-encoder with no change of spatial dimensions in feature maps throughout the layers, that comprises three convolutional layers. The first two layers are accompanied by ReLU activation layers. The convolutional filters are implemented with no padding on the input signals. As all the convolution filters have a stride of 1, the output signal is expected to have the same spatial shape as the input signal. However, lack of padding causes a slight reduction in the output size. The authors avoided inclusion of padding to overcome the border effect that might occur by introduction of zero-padding in the feature maps.

An important characteristic of SRCNN is that the actual up-scaling happens as a pre-processing step by performing bi-cubic interpolation. Consequently the input to the CNN has already the desired spatial resolution, therefore the CNN is essentially an image enhancement and de-noising platform that improves the visual consistency of already up-scaled image. Deep learning-based methods in SR can be categorized into

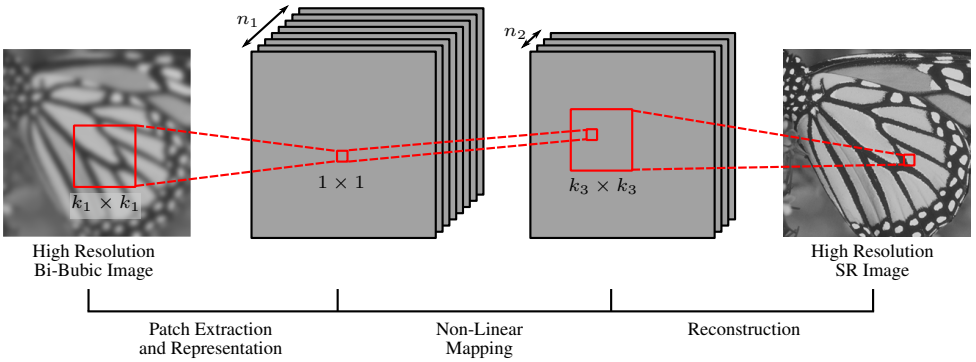


FIGURE 2.2: The architecture of the SRCNN model [30, 31] for performing still image SR.

two classes of architectures that use the bi-cubically up-scaled image as their inputs, and architectures that take the original low resolution images as input and perform the up-scaling within the architecture. SRCNN is amongst the former class of architectures.

Figure 2.2 depicts the schematic architecture of the SRCNN with three convolutional layers. The activation layers are not included in the figure, although the first two layers are accompanied by ReLU functions. According to the figure, the model is designed for single-channel image SR, i.e. only the grayscale images or the luma signal is up-scaled using the proposed architecture, and the color channels (if available) are up-scaled by other means, although the same model can easily be adapted to handle multi-channel inputs. The authors suggest a range of possible values for the different parameters in Figure 2.2, but the typical values are $k_1 = 9$, $k_3 = 5$, $n_1 = 64$, and $n_2 = 32$.

As noted in Figure 2.2, each layer of SRCNN has its own specific design goals. According to Dong et al. [30, 31] the first layer takes charge of patch extraction and representation. The second layer is a non-linear mapping, which takes an n_1-dimensional input and maps it to an n_2-dimensional vector. Finally the third layer is a reconstruction layer which performs a filtering on the final set of feature maps leading to generation of the output high resolution enhanced signal. This categorization of the three layers are inspired by the three general steps that are common in performing learning-based SR. In particular the authors provided an interesting analogy between their presented CNN-based architecture, and the sparse representation-based SR approach [129, 130], in which every layer corresponds to a step in the sparse coding-based models.

In terms of training, the Mean Squared Error (MSE) is chosen as the cost function, and a set of cropped images are used as the training data. Although SRCNN is a very simple deep learning-based approach with few convolutional layers, it can still

outperform the existing learning-based SR approaches of its time, and opens up a wide range of possibilities for developing and improving CNN-based approaches for both still image and video SR. SRCNN can be considered as the first pioneering work in CNN-based SR that inspired the future developments in this field. However, there are several other deep learning-based approaches that also considered application of CNNs for SR in different ways.

In that regard, Cui et al. [27] employed a cascade of multiple collaborative local auto-encoders that up-samples the low-resolution image gradually layer by layer. In each layer of the cascade, non-local similarity search is carried out first, then input patches are fed into a collaborative local auto-encoder for further enhancement. Unlike the SRCNN, this method takes the original low resolution input as the input and performs the scaling within the CNN architecture.

Wang et al. [121] introduced a deep joint SR model to exploit both external and self-similarities for reconstruction, in which a stacked de-noising convolutional auto-encoder is first trained on external training data, then it is fine-tuned with multi-scale self-examples from the input data.

In [120] and [81], the authors argued that the domain expertise from the conventional sparse representation models can be combined with the desirable features of the deep learning to achieve superior picture quality in SR. The resulting model is a cascaded deep learning architecture which subsumes the key notions of sparse coding, while providing satisfactory performance. The proposed scheme can be extended for handling various types of quality degradations such as noise and blurring artifacts.

Liu et al. [80] proposed the method of learning a mixture of SR inference components in a single inter-connected framework for performing SR. In particular, several SR inference modules tuned to different image local patterns are first applied independently on the input low resolution image to obtain a number of high resolution estimates, and the resulting estimates are adaptively aggregated and combined to form the final output image. Selecting neural networks as the SR inference module enables the entire procedure to be incorporated into a unified network and be optimized jointly.

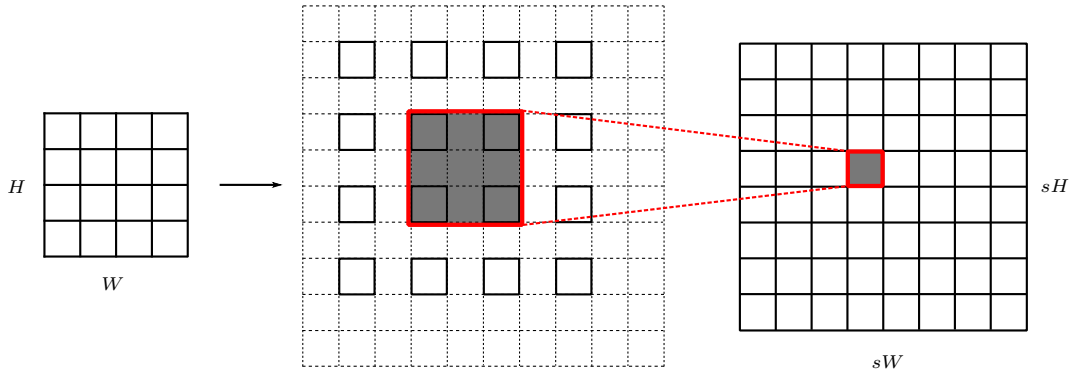


FIGURE 2.3: The visual presentation of transposed convolutional filtering. The dashed lines are the paddings applied to the input signal.

2.3.1 Transposed Convolution Layers for Up-Scaling

As mentioned earlier, CNN-based SR can be classified into two types of approaches. One group of approaches enhance the pre-processed bi-cubically up-scaled image to a higher quality version with the same resolution, whereas the other group perform the up-scaling within the CNN structure, using different trainable layers. A prominent method that took the direction of the latter approach was proposed by Dong et al. [32]. They improved the SRCNN model by introduction of transposed convolutional layers [134, 135], also know as deconvolution layers.

The transposed convolution can be regarded as an inverse convolution operation. In convolution, when a kernel is convolved with an image with a stride of size s, the dimensions of the resulting output is $1/s$ of the input image in each spatial direction. For transposed convolution, however, when a kernel is convolved with an image with a stride of size s, the dimensions of the resulting output are s times the input image in each spatial direction. In other words, transposed convolution with stride s can be considered as a normal convolutional layer with a stride of $1/s$ [100]. Figure 2.3 illustrates a visual representation of the transposed convolution filtering. In that example, the input signal is a $4 \times 4 \times 1$ image, and the filter has a 3×3 kernel. The transposed convolution is performed with $s = 2$, and enough padding is applied to achieve an output signal with its spatial dimensions twice as the input signal.

Employing transposed convolution layers will assist the network to recover the lost information, or enlarge the feature maps. Transposed convolution can be considered as

an up-sampling layer with filter kernels that are trainable. Therefore, Dong et al. [32] employed them as such to perform the image up-scaling within the CNN structure. The FSRCNN [32] method follows the same structure as SRCNN, with feature extraction and non-linear mapping stages within the CNN. The operations are, however, performed on the low resolution image, and there are no bi-cubic interpolation pre-processing step in the framework. The final stage of the CNN architecture, which deals with reconstruction, is concluded with a transposed convolution layer with a stride size equal to the desired scaling factor. This leads to restoration of the input image in the target high resolution.

One of the key motivations for employing transposed convolution layers by Dong et al. [32] was providing a low complexity version for SRCNN, which led to development of FSRCNN. The objective of employing the transposed convolution as the last layer of the CNN is to perform all the computational operations in the native low resolution of the input image, and perform the up-scaling in the final stage. This approach can reduce the complexity of the SR framework significantly, as performing convolutional filters on lower resolution images and feature maps requires less memory and power, leading to reduction in overall complexity of the system. Dong et al. [32] also improved the picture quality compared to the SRCNN by adopting larger training data sets and integrating more layers in the FSRCNN structure.

It is worth noting that in the context of SR, employing transposed convolution layers is not limited to FSRCNN, and it is further exploited in more complex network architectures. In particular, transposed convolution layers can be a key element in encoder-decoder architectures, and they are widely used in symmetric SR networks. Some of the existing SR encoder-decoder architectures invest in reducing the spatial size of the input and compressing its information by transferring it to another space in the first half of the network, and then recovering the information and reconstructing the output image by transferring compressed representations to the pixel space on the second half of the network. The first half of the process is similar to an encoding process, and the recovery stage is considered as a decoding process. The encoding stage is performed using normal convolutional layers (with different strides in this case), and the recovery stage can be performed using transposed convolutional layers (also with various stride sizes). The method proposed by Mao et al. [85] is an example, which will be described in more details in the following sections.

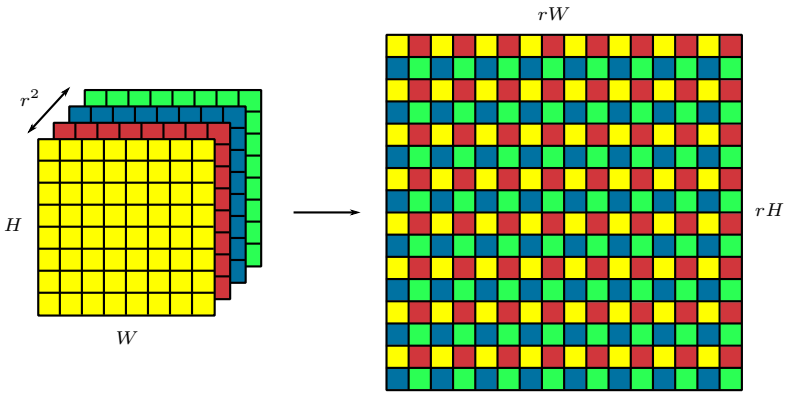


FIGURE 2.4: The periodic shuffling proposed in ESPCN model [101].

2.3.2 Sub-Pixel Up-Scaling

Performing the scaling operations within the CNN is an interesting challenge that has been addressed in different network architectures, not necessarily related to the SR models. The up-scaling process can be typically handled by un-pooling [133], or *perforate* [92]. Another efficient method that was specifically designed for SR networks is the sub-pixel convolution layers and periodic shuffling operation proposed by Shi et al. [101].

In [101], Shi et al. devised the ESPCN for image SR that is built upon the SRCNN model, and follows the path of FSRCNN in terms of performing the up-scaling in the final layer of the CNN, avoiding the bi-cubic interpolation as a pre-processing stage in SR. The up-scaling is performed by a sub-pixel convolution layer, which comprises a normal convolution layer accompanied by a periodic shuffling mechanism that rearranges the elements of a $H \times W \times r^2C$ tensor into a tensor with the shape $rH \times rW \times C$, with r representing the scaling factor and the sub-pixel operation parameter. This operation is illustrated in Figure 2.4 for the case of $r = 2$ and $C = 1$. The periodic shuffling operation $\mathcal{PS}$, can be formulated in mathematical terms as the following:

$$\mathbf{M}_{x,\ y,\ c} = \mathcal{PS}(\mathbf{T};\ x,\ y,\ c) = \mathbf{T}_{\lfloor \frac{x}{r} \rfloor,\ \lfloor \frac{y}{r} \rfloor,\ C \cdot r \cdot \bmod (y,r) + C \cdot \bmod (x,r) + c} \tag{2.3}$$

where $\mathbf{M}$ is the resulting output tensor, and $\mathbf{T}$ is the input tensor, with x, y, and c representing the pixel coordinates of $\mathbf{M}$. As sub-pixel convolution layer is a normal convolution operation coupled with the periodic shuffling process, the convolution operation in the sub-pixel layer requires to have r^2C filters, resulting in tensor $\mathbf{T}$ with r^2C channels.

Shi et al. [101] used a large data set for training the ESPCN and by selecting an adapting learning rate, they obtained a SR model, which is far more efficient than the baseline SRCNN architecture. Moreover, the picture quality is improved in ESPCN by means of employing tanh activation instead of ReLU, as well as adaptive training. It is also worth noting that application of the periodic shuffling can be avoided during the training process by pre-processing the data set, which can lead to a more efficient training phase for the model.

The efficient structure of periodic shuffling and its impressive performance in SR models led to its popularity in several successive models. Sub-pixel up-scaling is used widely in SR, and several notable contributions such as [72] and [77] rely heavily on application of periodic shuffling for enlarging the feature maps and achieving the desired target resolution in images.

2.3.3 Residual Learning in SR

Other major contributions in single image SR using deep learning have been done by Kim et al. [63]. In [63], the concept of residual learning was introduced to SR within a deep structure, which can lead to quicker convergence of the network and high visual quality in picture reconstruction.

Another interesting endeavor in devising deep learning models for image SR was done by Mao et al. [85]. They introduced a very deep hourglass-shaped CNN that includes multiple convolution and transposed convolution layers originally designed for image denoising and taking advantage of residual learning. The model, however, can be applied for SR and image up-sampling as well.

The VDSR model by Kim et al. [63] was perhaps the first SR model that incorporated residual learning in the simplest form. The baseline model depicted in Figure 2.5 was inspired by Simonyan and Zisserman work [103]. Unlike the SRCNN, where the exact copy of the input image is aimed to be reconstructed as the output, VDSR generates the disparity between the high resolution and low resolution (bi-cubically up-scaled version) images. In order to reconstruct the final output, the disparity signal is added to the input signal element-wise. In the disparity (residual) signal, most values are either zero

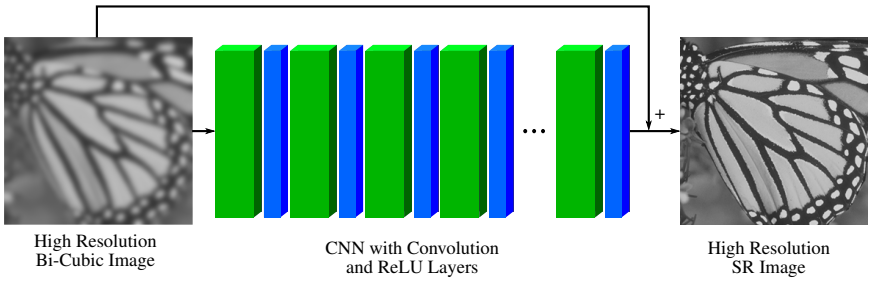


FIGURE 2.5: The residual learning concept applied with global skip connection in VDSR model [63].

or close to zero. This concept helps the network to experience a more efficient training cycle, and avoid vanishing and exploding gradient problems [16].

Mao et al. [85] took one step further, and employed the residual learning in various instances of the REDNet architecture proposed for image restoration. REDNet consists of an encoder-decoder architecture with multiple skip connections. Skip connections mean that the source of the connection is combined with the target of the connection in the network. In this case the combination is a simple element-wise addition of the signals. It is essential that the source and target signals in all skip connections have the same dimensions, so they can be added together. The reason for using skip connections is to have a better and smoother training process in the network. When the network becomes deep, transposed convolutional layers cannot fully recover the lost information in the network. In shallow networks with only a few layers, transposed convolution layers are capable of reconstructing the target signal.

In VDSR there exists one global skip connection. In REDNet depicted in Figure 2.6, which was inspired by deep residual networks [46], skip connections have been added between some of the corresponding convolutional and transposed convolutional layers that perform the resizing operations in the SR network. There are two reasons for employing skip connections. As mentioned earlier when the network becomes deeper, visual details can be lost, making it difficult for the transposed convolution layers to recover the information. However, the feature maps passed by the skip connections contain significant levels of information on visual details that can help the transposed layers to recover a better and higher quality image. Another reason for using skip connections is to achieve benefits on back-propagating the gradient to bottom layer, which makes training of the deep networks much easier.

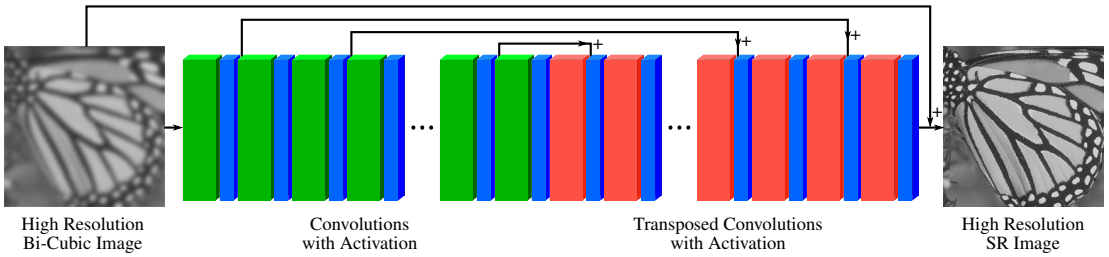


FIGURE 2.6: The general architecture of REDNet [85] model with skip connections.

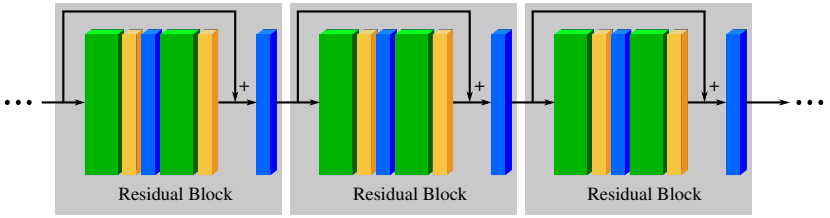


FIGURE 2.7: Residual blocks and local skip connections in advanced SR networks. Different colors are selected to represent the diversity of choices in designing a residual block.

Residual learning has been applied in a symmetric fashion in the REDNet, taking advantage of encoder-decoder models by employing transposed convolutional layers, in which the spatial shape of the feature maps change within the CNN. In more advanced architectures such as SRResNet proposed by Ledig et al. [72], the size of the feature maps remain constant in the network, and local skip connections are applied in addition to the global skip connections in a more adaptive and robust way using residual blocks. Residual blocks are a set of layers that are embedded in a virtual block with connected input and output, as depicted in Figure 2.7. There are different suggestions for selecting the right ingredients in residual blocks, and the positioning of activation layers and batch normalization, as well as connection considerations. However, the main notion is to cascade a series of residual blocks that form a very deep residual architecture, and can enhance the learning significantly according to He et al [46].

2.3.4 Generative Adversarial Networks for SR

The majority of the research in deep learning-based SR has been employing the auto-encoder architectures for designing accurate restoration models. An alternative CNN-based approach that can also promise high quality performance is based on Generative Adversarial Network (GAN) models. GANs, as an important branch of deep learning, are gaining more and more attention in the world. GANs were introduced by Goodfellow et al. [43] to model the training image data distribution, which is then used to generate new

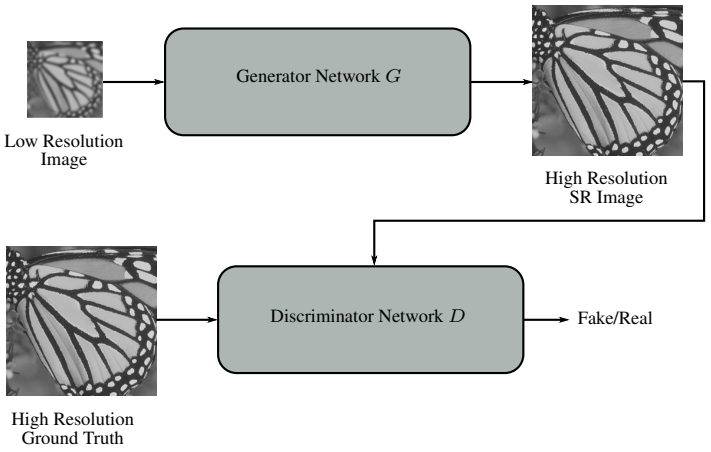


FIGURE 2.8: Schematic overview of GAN models with generator and discriminator networks.

image samples from the same distribution. Since their introduction, GANs have gained prominence as one of the most widely used methods for training deep generative models. A significant amount of investigations have been made to explore the potential of GANs in tasks related to natural images like image synthesizing [117], image compression [11], SR [72], and style transfer [138].

Ledig et al. [72] applied GANs in SR and provided an end-to-end adversarial network, named SRGAN, for generating photo-realistic high resolution images from low resolution content. Typical to GANs, the SRGAN model consists of generator G and discriminator D architectures, and they interact with each other during training as depicted in Figure 2.8. The generator network is a feed-forward auto-encoder that creates high resolution images and can be trained individually from the discriminator network if needed, which in turn will follow similar direction as the models described previously. The discriminator network is a classification model that decides whether the image created by the generator network is real or fake. The discriminator network is optimized by solving the adversarial min-max problem as formulated below. In Equation 2.4, $\mathbf{X}$ and $\mathbf{Y}$ represent the ground truth high resolution and the low resolution images, respectively. The optimization allows training of a generative model with the goal to fool the discriminator. This leads to a generator model that is capable of creating images that are highly similar and correlated to real world images.

$$J(D,G) = \mathbb{E}_{\mathbf{X}\sim p_{train}(\mathbf{X})}[\log(D(\mathbf{X}))] + \mathbb{E}_{\mathbf{Y}\sim p_{\mathbf{Y}}(\mathbf{Y})}[\log(1 - D(G(\mathbf{Y})))] \tag{2.4}$$

The first term acts as a counter for the correct decisions in identifying the original high resolution images as real images by discriminator, while the second term acts as a counter for the correct decisions in identifying the artificially generated high resolution images as fake. Solving $\min_G \max_D J(D, G)$ means minimizing the error for classification of artificially generated images as fake, while maximizing the accuracy of discriminator network. Ledig et al. [72] designed a deep CNN with residual learning as the generator network. The generator network can provide high quality SR images with favorable objective results when the network is trained independently from the discriminator and the adversarial training is avoided. Inclusion of adversarial learning leads to further improvements in the image quality and creation of more realistic textures in the reconstructed images. The objective evaluations, however, do not show any improvements when using adversarial learning, and the performance can only be assessed using subjective tests. This is one of the critical aspects of the GANs, that they may fail in standard objective evaluations, and their functionality depends heavily on extensive subjective tests that can be difficult to conduct.

The SRGAN method proposed by Ledig et al. [72] was further enhanced by Wang et al. [118]. By modification of the residual blocks in the generator architecture and study of the loss functions for network optimization, as well as employing the relativistic GAN [57] approach, Wang et al. [118] managed to improve the SRGAN in terms of picture quality. Application of GANs in SR is studied further in recent years and more contributions have been made in that regard. Yuan et al. [131] deployed an unsupervised learning method for SR. Using GANs as the core component, they proposed a cycle-in-cycle network structure, which removes the noisy and blurry artifacts in the first step, then the content is up-scaled to the target resolution. Wang et al. [119] developed a model for still image SR that has a progressive architecture and learning mechanism. The network performs the up-sampling in intermediate steps, and takes advantage of a discriminator architecture to create photo-realistic images.

2.3.5 Further Developments in Still Image SR

In previous sections, the evolution of deep learning-based still image SR was presented by outlining some of the key contributions in this field. This section summarizes some

further developments that rely on the work previously described with references to some of the recent publications in still image SR.

In [64], a deep and recursive architecture was employed, that can achieve very good performance in terms of image quality, although expensive in terms of computation cost. In the proposed framework, a deep recursive layer is embedded in a CNN architecture, which utilizes a very large context and allows for wide overall receptive field in the network. The network is further enhanced by introduction of recursive-supervision and skip connections in the learning process.

Tai et al. [108] deployed a similar approach as in [64], and presented a recursive architecture for still image SR. However, they took one step further and introduced both local and global residual learning to the recursive architecture to improve the learning process of the network, and achieve a more stable model given the large number of parameters. The trained model surpasses the model presented in [64] in terms of picture quality.

Lim et al. [77] proposed the EDSR model, which is an enhanced deep residual network for still image SR inspired by the generator network of SRGAN presented by Ledig et al. [72]. The improved performance of the EDSR is thanks to the advanced design choices including devising the right components for residual blocks, expanding the model size, and stabilization of the training process. EDSR is discussed in more details in Chapter 5. Lim et al. [72] also introduced a unified multi-scale network that can perform SR for more than one scaling factor.

Tong et al. [113] also provided a deep residual architecture with more elaborate residual blocks. Introduction of dense skip connections leads to propagation of the feature maps from each layer into all subsequent layers, providing an effective mechanism for combination of low level and high level features. The model employs transposed convolution layers for performing up-scaling, and a favorable picture quality is achieved by the proposed method.

Zhang et al. [139] presented a deep residual dense architecture for still image SR that exploits the hierarchical features from all the convolutional layers. The authors presented residual dense blocks for extracting abundant local features via densely connected convolutional layers. Local feature fusion is also used in the residual dense blocks to learn

more effective features adaptively from previous and current features and having a stabilized training process. The model is complemented with incorporation of global feature fusion for jointly learning the global hierarchical features, leading to improved picture quality in the reconstructed high resolution images.

Ahn et al. [12] adapted a progressive learning scheme to the deep convolutional networks. In a nutshell, the end-to-end training of the model is performed in multiple stages, so the model can gradually increase the output image size. The network comprises cascading residual blocks with local and global skip connections, and the model can promise high quality image reconstruction, competing with EDSR picture quality.

In [102], Shocher et al. presented an unsupervised SR model that takes advantage of internal recurrence of information within a single image and train a small image-specific CNN at inference time, using the examples and instances extracted solely from the test image. This leads to image-specific models that are fine-tuned to the data in question, and can easily be adapted to handle any type of distortions and artifacts existing in image restoration scenarios such as recovering information from old footage and noisy archive material.

2.4 Deep Learning-Based Multi-Frame Super-Resolution

There have also been efforts in applying deep learning in multi-frame SR. In principle, multi-frame SR is an extension of still image SR, where the correlation between several overlapping frames are exploited to reconstruct a high quality high resolution target frame from the set of input images. Deep learning-based multi-frame SR models are developed after the effectiveness of CNNs were demonstrated in still image SR. Inspired by the advances in deep learning and the prior knowledge in multi-frame SR, several contributions in deep learning-based multi-frame SR have been made in recent years.

In general, majority of multi-frame SR models consist of two key building blocks. The first building block is in charge of aligning the input frames. Frame alignment is a critical task that involves analysis of the temporal behavior within the input frames and it typically requires motion estimation and flow analysis methods to be able to create spatially aligned frames that have significant correlation with each other. This leads to better representation of the features in the next building block, which handles the

SR and up-scaling operations, as well as quality enhancement if needed. Multi-frame SR is often applied on video signals, as videos consist of consecutive frames with high correlation. Hence video SR is also a common term that is used interchangeably with multi-frame SR. Figure 2.9 illustrates a simple representation of the multi-frame SR concept with the two building blocks. The input low resolution frames are labeled as $\mathbf{Y}_i$ and the output high resolution frames are labeled as $\mathbf{X}_i$, with i representing the temporal index of the frames. The output sequence is generated frame-by-frame in majority of existing solutions, and multiple inputs can participate in generation of each output high resolution frame. In some of the proposed approaches the previously generated high resolution frames can also participate in generation of the next output frame, as shown with dashed lines in Figure 2.9. In the following, several pioneering contributions in multi-frame SR using CNNs are outlined.

Liao et al. [76] applied deep learning in multi-frame SR by using a two-step process, in which a SR draft ensemble generation and its optimal reconstruction take place. In the first step, two motion compensation algorithms are used to calculate SR draft in order

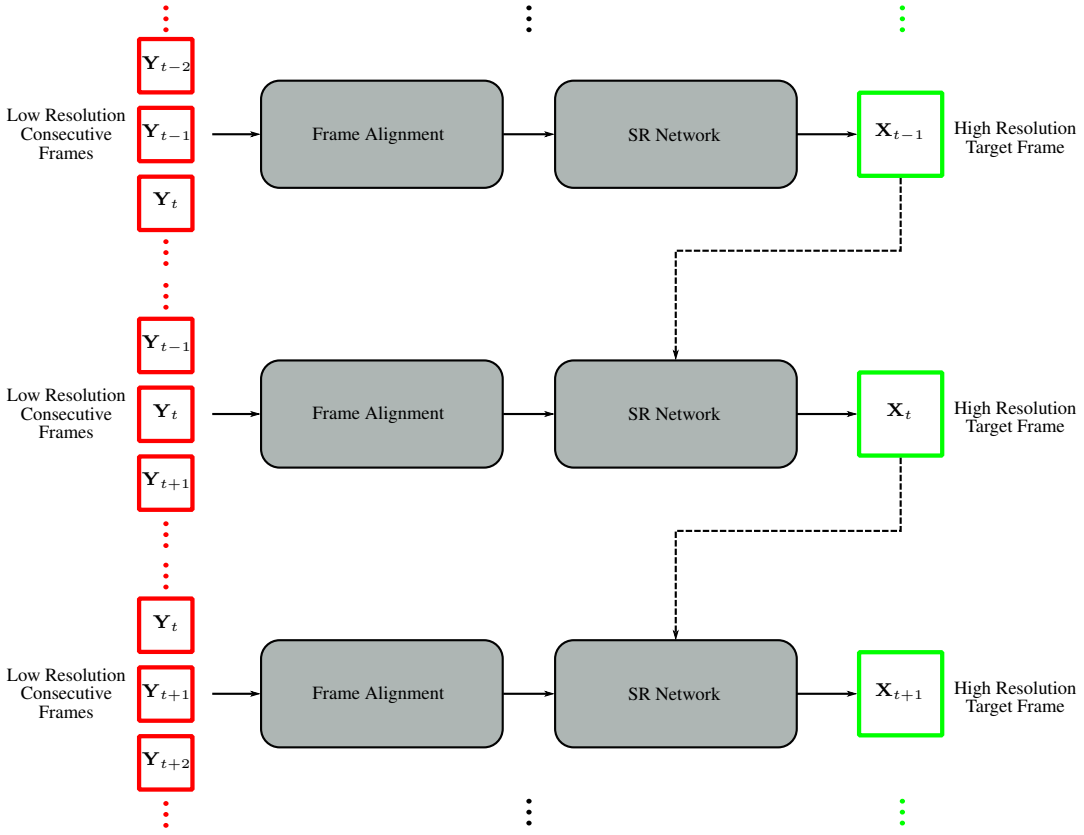


FIGURE 2.9: Schematic overview of multi-frame SR networks. The details and connections may vary for each specific model.

to deal with motion compensation errors. In the next step, all drafts are combined using a CNN. This method is computationally expensive and also leads to a frame-by-frame generation of the high-resolution video eventually, which is not very efficient.

Huang et al. [49, 50] introduced an interesting approach, which is actually a direct mapping between multiple low-resolution frames to the same number of high resolution frames, using a bidirectional recurrent CNN. This approach exploits the temporal information within the frames and does not require a motion compensation step, which is a typical case for multi-frame approaches. Although very effective, there is a bi-cubic up-scaling pre-processing step in the pipeline, which results in extra complexity to the framework, as it causes the network to do all the processes in high resolution domain.

Kappeler et al. [61] presented a multi-frame SR approach to the original CNN architecture introduced by Dong et al. [30]. Their approach improves the performance of SR with respect to the single image architecture, but the presented architecture cannot fully exploit the temporal information, as the frames are implemented as separate input channels in the network input, and the channels collapse into one channel at the output, causing the loss of temporal information during the course of learning. Moreover, the model does not contribute to the efficiency of video SR, as the output is a single frame. The method depends heavily on a motion compensation process which increases the complexity.

Makansi et al. [84] presented a multi-frame SR approach, which is very similar in nature to the model presented by Kappeler et al. [61]. However, an end-to-end deep learning-based motion compensation framework based on optical flow estimation is employed, which can be coupled with the SR module and boost the performance of the multi-frame architecture. One benefit of this approach is that the entire framework comprising of the motion compensation module and the SR module can be trained together simultaneously, which can create a better learning condition for the SR.

Caballero et al. [23] also presented an end-to-end trainable multi-frame SR network, that encompasses the motion compensation and enhancement stages in one architecture. They made use of an earlier still image SR [101] concept, and incorporated the sub-pixel scaling layer in both motion compensation and enhancement sub-networks to create an end-to-end solution for multi-frame approach that results in efficient and high quality performance.

One of the interesting works in multi-frame SR was done by Tao et al. [109]. Their CNN-based multi-frame approach is designed by building on previous work and introduction of a sub-pixel motion compensation layer, that can incorporate the motion compensation and similar frame fusing concepts, and create an end-to-end trainable framework for multi-frame SR.

Liu et al. [79] proposed a temporal adaptive deep learning model that can adaptively identify the temporal dependency scale and handle various types of motion and alleviate the effect of detrimental impact of erroneous motion estimation between adjacent frames. Moreover, the approach utilizes a spatial alignment network that can robustly and efficiently perform low complexity frame alignment. The approach results in an end-to-end trainable architecture that combines the existing steps in one joint learning framework.

In [94], Sajjadi et al. proposed an end-to-end trainable frame-recurrent video SR model that exploits the previously reconstructed high resolution frames to up-scale the subsequent frames. The model is an actual video SR framework, and differs from the typical multi-frame approaches, in that estimation of the high resolution frames is not done independently for each target frame and the preceding estimations play an important role in the framework. The approach encourages temporally consistent results and computationally efficient performance. Moreover, due to the recurrent nature of the model, a large number of previous frames can be incorporated in reconstruction process of target frames, which leads to higher picture quality.

Jo et al. [55] took a rather different approach in designing a deep learning-based video SR framework. An end-to-end deep learning architecture was proposed, that generates dynamic up-sampling filters and a residual image computed based on the local spatio-temporal neighborhood of each pixel, avoiding explicit motion compensation. In this scenario, a high resolution frame is reconstructed directly from the input low resolution frame by dynamic up-sampling filters, and the accuracy of the image estimation is enhanced by the computed residual.

An important point about most of the multi-frame SR approaches is that even though they are often identified as video SR solutions in the literature, they often generate the final output on a frame-by-frame basis, except for a few cases in existing art. The main notion of multi-frame SR approaches is that multiple neighboring frames from

a video sequence are taken and one single target frame is produced as the output. The neighboring frames are essentially assisting the SR mechanism to produce a higher quality target frame by providing more textural information to the framework. This, however, adds to the complexity of the framework and introduces latency in processing of the videos, as most of the approaches require a motion compensation step, and the final high resolution video generation is still on a frame-by-frame basis.

2.5 Quality Metrics for Evaluation of Super-Resolution

There are several objective metrics that are widely used in evaluation of the SR algorithms. The most prominent ones are Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) index, which compare two images, i.e. an artificially generated image X^* and the ground truth X, and compute a score that represents the similarity of the images.

PSNR is based on the MSE function and can be formulated as:

$$PSNR(X^*, X) = 20 \cdot \log_{10}(\frac{M^2}{MSE(X^*, X)}) \tag{2.5}$$

with M representing the highest value in the pixel dynamic range, which is typically 255 in an 8-bit pixel representation scheme. MSE can be formulated as:

$$MSE(X^*, X) = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} (X^*{}_{i,j} - X_{i,j})^2 \tag{2.6}$$

with W and H representing the width and the height of the images. A high value of PSNR corresponds to high similarity between the two pictures, while the minimum value for the PSNR can be zero.

The other well-known method for quality assessment in SSIM. The difference between SSIM and PSNR is that PSNR measures absolute errors, while SSIM is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close. These

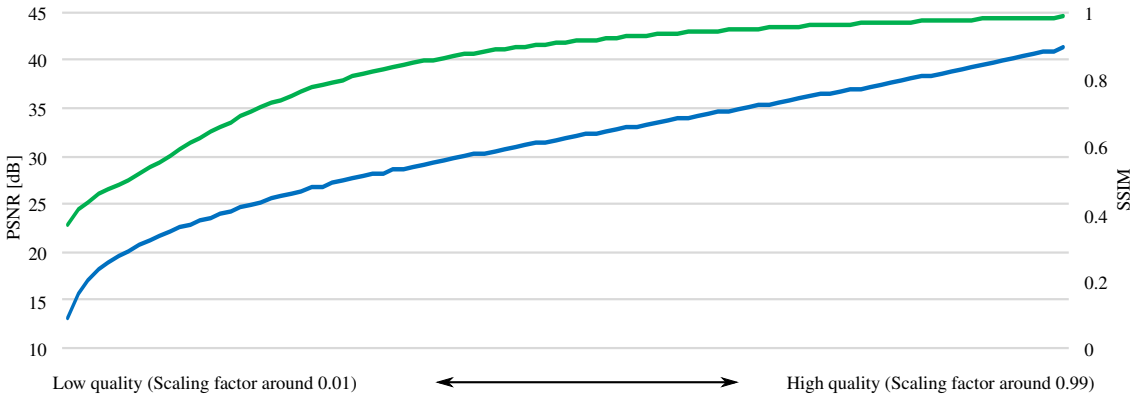


FIGURE 2.10: The behavioral analysis of PSNR (blue) and SSIM (green) over a range of content from low to high quality.

dependencies carry important information about the structure of the objects in the visual scene. Luminance masking is a phenomenon whereby image distortions tend to be less visible in bright regions, while contrast masking is a phenomenon whereby distortions become less visible where there is significant activity or texture in the image. SSIM can be formulated as:

$$SSIM(X^*, X) = \frac{(2 \cdot \mu(X^*) \cdot \mu(X) + c_1) \cdot (2 \cdot \sigma(X^*, X) + c_2)}{(\mu^2(X^*) + \mu^2(X) + c_1) \cdot (\sigma^2(X^*) + \sigma^2(X) + c_2)} \tag{2.7}$$

where μ is the mean value function, $\sigma(X^*, X)$ is the covariance function, and $\sigma^2(X^*)$ and $\sigma^2(X)$ are the variance functions. c_1 and c_2 are two stabilization variables defined as:

$$c_1 = (k_1 \cdot M)^2 \tag{2.8}$$

$$c_2 = (k_2 \cdot M)^2 \tag{2.9}$$

with $k_1 = 0.01$ and $k_2 = 0.03$ by default. SSIM is between -1 and 1, while a value of 0 represents no structural similarity, and a value of 1 occurs only in case of having two identical images.

The following experiment is a simple way of demonstrating the ranges of PSNR and SSIM and their behavior in analyzing the picture quality between a generated image and the ground truth. In this experiment five images of the Set 5 data set are initially down-scaled, then up-scaled using bi-cubic interpolation. The following 99 scaling factors $\{0.01, 0.02, 0.03, ..., 0.99\}$ where used for down-sampling, and up-sampling was performed to create an image with the original resolution. The outcomes were compared with the ground truth data, and the average values of PSNR and SSIM for every factor were

recorded. Figure 2.10 summarizes the results, showing a wide range of outcomes for PSNR and SSIM, while representing very low quality cases around the left side (scaling factors around 0.01), and very high quality cases around the right side (scaling factors around 0.99) of the graph.

2.6 Summary of Methods in Super-Resolution

Table 2.1 summarizes the key contributions in development of SR methods for still images and videos based on different approaches, and in particular deep learning.

TABLE 2.1: Summary of key contributions in SR.

Single image	Model-based	Joint MAP registration [44] Markov random fields [59] Variational Bayesian model [14]
	Conventional learning	Self-similarities approach [35, 48, 102, 127] Sparse representation [128–130, 136] Neighbor-embedding [24, 111] Random forests [95, 99, 142]
	Deep learning	Baseline: SRCNN [30, 31] Early models [27, 80, 81, 120, 121] Low complexity: ESPCN [101], FSRCNN [32] Residual learning: VDSR [63], REDNet [85], EDSR [77] Generative adversarial networks [57, 72, 118, 131] More recently [12, 113, 139]
Multi-frame	Model-based	Maximum a Posteriori [15] Adaptive Bayesian [78]
	Conventional learning	Self-similarities approach [35] Sparse representation [28, 62, 104, 114] Neighbor-embedding [112]
	Deep learning	Bidirectional recurrent CNN [49, 50] SRCNN extension [61] End-to-end motion estimation and SR [23, 84, 109] More recently [55, 79, 94]

Chapter 3

Fast Training of Super-Resolution Convolutional Networks

In this chapter, a novel cost function is presented for fast and accurate training of deep learning-based Super-Resolution (SR) models, that can replace the widely used Mean Squared Error (MSE) function. The proposed cost function is designed based on the existing visual quality metrics, and has unique characteristics that can lead to efficient and fast convergence of neural networks. The functionality of this approach is validated by training state-of-the-art SR convolutional networks, and achieving significant reductions in training time of some of the well-known encoder-decoder architectures. The effectiveness of the method in SR task also promises similar improvements for other image and video restoration tasks that depend heavily on application of generative architectures.

3.1 Introduction

The majority of deep learning-based restoration models are based on application of encoder-decoder architectures, in which the input signal (low resolution/quality image) is mapped to a different space with a different dimensionality using an encoder network, and the decoder acts as a reverse mapping that generates the output signal (high resolution/quality image) in the original space. Auto-encoder models are a variant of the encoder-decoder architectures, which are being used widely for generative models, as well as content restoration and SR.

With regard to the state-of-the-art approaches in training SR networks, MSE is widely used as the cost function, providing favorable results in terms of quality for trained models. However, the behavior of the MSE function, along with its inconsistency with available image and video quality metrics, leads to inefficient and time-consuming training of restoration models, and particularly SR networks.

In this chapter, an alternative cost function is presented, that has a more meaningful structure and is more adapted to the visual quality metrics widely used in evaluation of SR models. However, the greatest advantage of the proposed loss function is its impact on speeding-up the training process and reducing the number of back-propagations needed for convergence of SR models. The application of the proposed cost function leads to major speed-ups in training SR models, while retaining the same level of quality as the models trained with classic MSE.

The rest of this chapter is organized as follows. Section 3.2 describes the related work and state-of-the-art approaches in single image SR and training Convolutional Neural Networks (CNN) for image restoration. Section 3.3 provides a brief description of the training procedure and cost function optimization in state-of-the-art deep learning-based SR models that utilize encoder-decoder structures. Section 3.4 provides details on designing an efficient cost function for training SR neural network models. Section 3.5 reports on the systematic evaluation of the proposed method, followed by the conclusions in Section 3.6.

3.2 Background

The growing popularity of deep learning and CNNs in recent years along with advances in GPU technology that made training these models possible, caused major breakthroughs in various machine learning and computer vision tasks. SR is not an exception, and application of CNNs and deep learning concepts for spatial up-sampling of images and videos has become a substantial research topic in recent years. Chapter 2 provided an extensive review of the existing approaches in deep learning-based SR.

The still image SR methods mentioned in previous chapter cover the state-of-the-art architecture designs and the two generic approaches of encoder-decoder architectures and adversarial approaches. With the exception of adversarial approaches such as [72],

that employ generative adversarial networks (GAN) for SR, the rest of the methods make use of encoder-decoder or auto-encoder concepts for performing high quality image up-sampling and MSE (or ℓ_2-norm) is the unified choice as the cost function for optimization. In addition to the GAN approach [72] that made use of the VGG [103] loss by building upon previous work in generative models [21, 38, 56], there are other studies in cost function optimization, and efforts in finding alternatives for MSE.

In [68, 69], the authors developed a deep Laplacian pyramid network, with a cost function based on the Charbonnier penalty[1]. Sajjadi et al. [93] also took advantage of an auto-encoder structure for single image SR, however they used additional terms in the cost function for incorporating a perceptual loss in feature space, as well as texture matching loss between the network output and the ground truth. Additionally, Zhao et al. [140] performed a comprehensive analysis on performance of several cost functions for image restoration using neural networks, in addition to proposing a cost function that can promise superior quality performance in comparison with MSE.

Although the above contributions do a good job in proposing novel cost functions and comparing the alternative approaches, the sole focus in all previous studies has been on quality of the image, and the efficiency of the training process and the number of needed back-propagation iterations for convergence of the neural networks have been overlooked and of minor importance in recent studies in image SR and restoration. That was the inspiration for this chapter to carry out investigations on alternative solutions for reducing the training time of the SR networks and provide a cost function that can guarantee fast and efficient training of SR encoder-decoder models, while providing high quality image reconstruction comparable with models trained with MSE.

3.3 Training Super-Resolution Networks

SR networks are trainable CNNs that learn a correspondence between low resolution and high resolution images based on the training data served to the network. If $\mathbf{Y}$ represents the low resolution image as a matrix (or tensor), and $\mathbf{X}$ represents the associated high

[1] Charbonnier loss is a smoothed from of ℓ_1 loss that behaves like ℓ_2 near the origin, and like ℓ_1 elsewhere, also known as pseudo-Huber loss.

resolution image, then the following equation formulates the SR network behavior:

$$\mathbf{X}^* = \theta(\mathbf{Y}) \tag{3.1}$$

where $\mathbf{X}^*$ represents an approximation of $\mathbf{X}$, and θ models the end-to-end representation of the CNN, subsuming all the existing parameters within the network. It is also worth mentioning that depending on the structure of the network, $\mathbf{Y}$ can be either the original low resolution image, or an up-sampled version of that image obtained by interpolation.

In training SR networks, a set of N coupled low and high resolution images are required. During the training phase, the SR network is exposed to the training data pairs, as batches of pairs. Batches are a sub-set of training set that are selected randomly, fed to the network to undergo the training optimization. Batches, or mini-batches of training set are used as opposed to the entire data set in one go to increase the efficiency and generalization of the training process, in particular for the case of gradient-based algorithms. The size of the batches are typically defined based on the available hardware resources for training.

During the training, the input low resolution images are processed by the network, and high resolution output images are generated. The network minimizes the disparity between the produced high resolution images and the original high resolution training data, also known as the label data, by modifying the values of kernels and biases within the network. The network parameters eventually converge to a set of values, which can produce a satisfactory generative model for SR.

Cost function is a critical element in training neural networks, which measures the error between the labels and the network outputs. Given that SR networks generate images as their output, a natural choice for cost function is the MSE, which is widely used for training not only SR networks but also other types of learning applications. In principal, MSE compares the content of two images (matrices) in pixel domain, and as a cost function, it can be represented as:

$$J_{MSE}(\theta; \mathbf{X}, \mathbf{Y}) = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} \left(\mathbf{X}_{i,j} - \theta(\mathbf{Y})_{i,j}\right)^2 \tag{3.2}$$

with W and H representing the width and the height of the target images. Minimizing (3.2) results in convergence of the network and learning an SR model. Application of

MSE as the cost function, may lead to poor performance in gradient-based optimizations, and although very popular in SR domain, it often results in slow convergence of the CNNs. Generally in machine learning, some output units that have saturatation issues produce small gradients when combined with MSE. Consequently, the gradient can shrink too small to be applicable for learning, resulting in ineffectiveness and deficiency of MSE [42].

This inspired the research presented in this chapter to analyze MSE in state-of-the-art SR models, and devise an alternative cost function for training the CNNs in the SR task. The next section describes the proposed approach in detail.

3.4 Efficient Cost Function Design

In designing the cost function for learning SR models, two important aspects need to be considered. The first aspect is the correlation of the cost function with the quality metrics in that domain, as well as soundness of the function in terms of quantification of the error. The second aspect, which is common in any other learning task, is devising a cost function that can provide a quick and accurate learning and convergence for the network.

3.4.1 Proximity Cost

In order to address the first aspect of the cost function design, existing visual similarity and objective quality metrics are considered, in addition to the MSE which is already a meaningful cost function. The first function to consider is the correlation between the network outputs and labels defined as the following:

$$\rho(\theta; \mathbf{X}, \mathbf{Y}) = \frac{\sigma(\mathbf{X}, \theta(\mathbf{Y}))}{\sigma(\mathbf{X}) \cdot \sigma(\theta(\mathbf{Y}))} \tag{3.3}$$

where the numerator represents the covariance and the denominator represents the multiplication of the standard deviation variables. The above equation measures the similarity (or proximity) of the network output and the label data, and a high value of the function corresponds to high similarities and low disparities between the two signals. Therefore maximization of the correlation function over the network parameters would

theoretically lead to the convergence of the network. This makes more sense when looking at the definition of one of the popular visual quality metrics, Structural Similarity (SSIM) index, defined in the following:

$$SSIM(\theta; \mathbf{X}, \mathbf{Y}) = \frac{(2 \cdot \mu(\mathbf{X}) \cdot \mu(\theta(\mathbf{Y})) + c_1) \cdot (2 \cdot \sigma(\mathbf{X}, \theta(\mathbf{Y})) + c_2)}{(\mu^2(\mathbf{X}) + \mu^2(\theta(\mathbf{Y})) + c_1) \cdot (\sigma^2(\mathbf{X}) + \sigma^2(\theta(\mathbf{Y})) + c_2)} \tag{3.4}$$

where μ is the mean value of the matrices and c_1 and c_2 are constants. As high SSIM values correspond to high similarity between two images, and given that both covariance and standard deviation (variance) values appear in the definition of the SSIM, these functions seem as logical candidates for the cost function. Based on these observations, a *proximity* cost is defined which is dependent on the covariance between the network outputs and the labels, and the standard deviation of the network outputs. The following is the definition of the proposed proximity cost function:

$$J_P(\theta; \mathbf{X}, \mathbf{Y}) = \frac{(\sigma(\mathbf{X}) - \sigma(\theta(\mathbf{Y})))^2}{\sigma(\mathbf{X}, \theta(\mathbf{Y}))} = \frac{\sigma^2(\mathbf{X}) + \sigma^2(\theta(\mathbf{Y}))}{\sigma(\mathbf{X}, \theta(\mathbf{Y}))} - \frac{2}{\rho(\theta; \mathbf{X}, \mathbf{Y})} \tag{3.5}$$

Studying equations 3.3 and 3.4 shows that maximization of those functions can result in convergence of the image pairs, and that corresponds to high values for the covariance and low values for the variance. In the deep learning optimization, often a function with minima is of interest, hence the proposed proximity function has an inverted behavior compared to the correlation function and SSIM, in that the standard deviation of the network output is in the numerator and the covariance of the labels and network outputs are in the denominator.

From a semantic perspective, a low value of numerator in the proximity function refers to a low disparity between the standard deviations of the network outputs and the labels, which accounts for similarity of the signals. For the case of exact similarity between the network outputs and labels, the numerator will be zero. Moreover, a high value of covariance in the denominator of the proximity function implies the closeness of the network outputs and the label signals.

3.4.2 Logarithmic MSE

In addition to the above-mentioned cost function, MSE is also incorporated in the proposed approach by being combined with the proximity function. However, instead of

taking the MSE as it is, its logarithm is used, to provide a quicker convergence and optimization process, and address the second aspect of cost function design. Due to the steepness of the logarithm function, a quicker convergence is expected when using it as the cost function, and the common issue of early saturation and small gradients can be avoided. The application of logarithm function is also aligned with the Peak Signal-to-Noise Ratio (PSNR) metric, which is the most common visual quality metric, defined as the following:

$$PSNR(\theta; \mathbf{X}, \mathbf{Y}) = 20 \cdot \log(M) - 10 \cdot \log(J_{MSE}(\theta; \mathbf{X}, \mathbf{Y})) \tag{3.6}$$

where M represents the highest value in the pixel dynamic range. As shown in the above equation, $\log(J_{MSE})$ also appears in the PSNR calculations, hence making it a more suitable and natural candidate for the cost function, when compared with pure MSE.

As mentioned earlier in the chapter, Zhao et al. [140] performed a comparison of several cost functions for training of SR deep learning models, and the comparisons covered PSNR function as well. However, the focus of the analyses was the inference quality, and the training speed and complexity of convergence were ignored. Application of logarithmic MSE, inspired by the PSNR formulation, flattens out the maxima in the MSE function, leading to reduction in average curvature and quicker convergence. Moreover, the experiments show that the logarithmic value never reaches to negative infinity. This is due to the fact that the flat training samples are discarded from the data set. Furthermore, probability of having identical patches in training is close to zero based on the experiments carried out.

3.4.3 Modified Proximity-based Cost Function

To incorporate both MSE and proximity function in the cost, the following linear combination is applied:

$$\boxed{J_{MPC}(\theta; \mathbf{X}, \mathbf{Y}) = \log(J_{MSE}(\theta; \mathbf{X}, \mathbf{Y})) + J_P(\theta; \mathbf{X}, \mathbf{Y})} \tag{3.7}$$

The first term is logarithmic, posing a stronger impact on the combined loss, which is desirable as it capitalizes on the MSE. Several weightings were considered for the

second term ranging from 0.1 to 10. The < 1 weights showed instability in training in some experiments, and the > 1 weights slowed down the training process. Hence 1 was selected as the suitable weight for the proximity function according to the tests. The above cost function is named the Modified Proximity-based Cost (MPC) function, and it can be minimized using different optimization algorithms such as gradient descent. As the parameters of the network start converging, $\theta(\mathbf{Y})$ approaches to $\mathbf{X}$. That leads to a steep descent in the value of the first logarithmic term of the MPC. As for the second term of MPC, the convergence of the parameters, along with closeness of the $\theta(\mathbf{Y})$ and $\mathbf{X}$, result in a descending behavior in the numerator and an ascending behavior in the denominator. Consequently, the second term also has an absolutely descending behavior during the training phase, and the objective of the training would be to reach to the lowest possible value for MPC as the network keeps learning. The gradient of the proposed cost function is as follows:

$$\begin{aligned}\nabla J_{MPC}(\theta;\mathbf{X},\mathbf{Y}) &= \nabla \log(J_{MSE}(\theta;\mathbf{X},\mathbf{Y})) + \nabla J_P(\theta;\mathbf{X},\mathbf{Y}) \\ &= \frac{\nabla J_{MSE}(\theta;\mathbf{X},\mathbf{Y})}{J_{MSE}(\theta;\mathbf{X},\mathbf{Y})} \\ &+ \frac{\sigma(\mathbf{X},\theta(\mathbf{Y}))\cdot\nabla(\sigma(\mathbf{X})-\sigma(\theta(\mathbf{Y})))^2}{\sigma^2(\mathbf{X},\theta(\mathbf{Y}))} \\ &- \frac{(\sigma(\mathbf{X})-\sigma(\theta(\mathbf{Y})))^2\cdot\nabla\sigma(\mathbf{X},\theta(\mathbf{Y}))}{\sigma^2(\mathbf{X},\theta(\mathbf{Y}))}\end{aligned} \tag{3.8}$$

As presented above, the covariance between the network output and the labels appears in the denominator of the proximity cost function, as well as its gradient. One issue that may arise is that in theory this value can be zero. However, this can be easily avoided by removing the training samples with no pixel variation that represent DC planar images. Those samples do not make any contributions to the training process, as they contain no textures or edges, and can be easily discarded by checking their variance.

Another rare case leading to a zero covariance in the denominator of the proximity function may happen in the first iterations of the training by the randomly initialized weights and biases in the CNN. The initial network parameters can result in a condition,

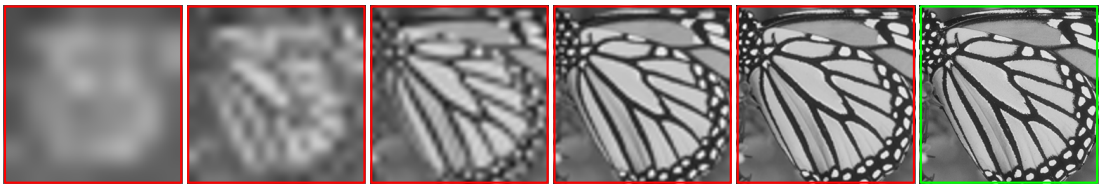


FIGURE 3.1: Degrading an image to different degrees of d_i: Degradation is performed by down-scaling the *Target* image by a factor of i and re-scaling it back to the original resolution.

TABLE 3.1: Typical values of MSE and MPC terms for the examples in Figure 3.1. The numbers represent different loss values between the degraded images and the *Target*.

	$\mathbf{d_{32}}$	$\mathbf{d_{16}}$	$\mathbf{d_8}$	$\mathbf{d_4}$	$\mathbf{d_2}$	**Target**
J_{MSE}	0.0505	0.0414	0.0224	0.0083	0.0024	0
$\log(J_{MSE})$	-2.9875	-3.1836	-3.8008	-4.7952	-6.0215	$-\infty$
J_P	2.0992	0.8278	0.1552	0.0186	0.0020	0

where estimated covariance value is zero. Although this can rarely happen, such scenario can be easily identified in the first iterations of the training and quickly dealt with by reinitialization of the weights and restarting the training process.

3.4.4 Qualitative Analysis of MPC

In order to have a better understanding of the proposed cost function, a simple example can be used to demonstrate the way different terms of the MPC act. Figure 3.1 shows a simple case of an image that has gone through different levels of degradation to simulate how a network can gradually converge and improve the reconstruction process of an image in a restoration model. The degradations of the *Target* image in Figure 3.1 are achieved by down-scaling the image by a factor of i and re-scaling it back to the original resolution by bi-cubic interpolation. Diverse levels of degradation ($i = \{2, 4, 8, 16, 32\}$) are considered to have a good mixture of distorted data. For every degraded version, three loss values are computed that measure the disparity between the degraded images and the *Target*. The MSE, as well as its logarithm and proximity function (that construct the MPC) are calculated and reported in Table 3.1. This simulation is a trivial way of showing how the two terms in the MPC behave, and how the speed and steepness of their alteration compared to the MSE can promise a much faster learning process for restoration models; a claim which is further investigated and proved in the next section.

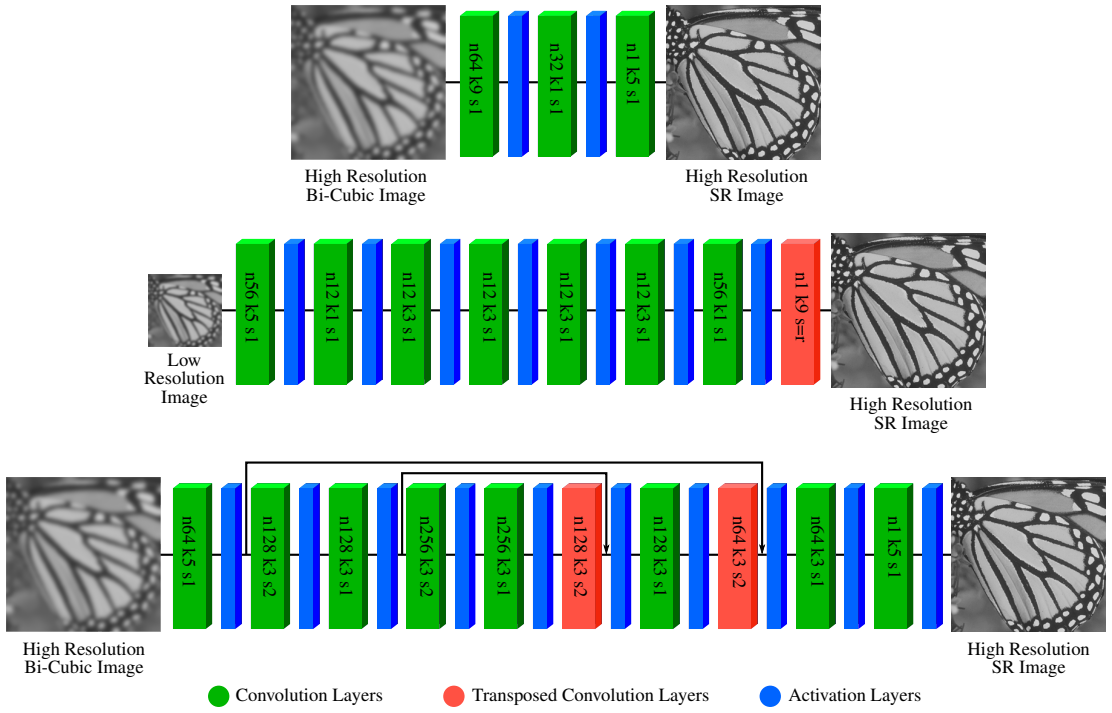


FIGURE 3.2: The three network architectures used for evaluations. From top to bottom: SRCNN [30], FSRCNN [32], and REDNet [85]. The parameters n, k, and s specify the number of filters, kernel size, and stride value for every layer, and r represents the scaling factor.

3.5 Experiments

The proposed cost function was tested for training three state-of-the-art still image SR networks and was compared in terms of performance with the conventional MSE cost function. Models trained by MSE are the sole baseline for comparative experiments in this study given that the main focus of this study is speeding-up the training process, while achieving the same quality level as models trained with MSE.

The three considered frameworks, depicted in Figure 3.2 include SRCNN [30], FSRCNN [32], and a variation of REDNet [85]. The selected models, apart from being amongst the most well-known and important contributions in SR field, represent three different approaches in designing CNNs for image restoration.

The SRCNN structure is based on the simple flat architecture defined as SRCNN-9-1-5 in [30], that takes the input image, already up-sampled to the target resolution by bi-cubic interpolation, and enhances it to a higher quality version with the same resolution. The model represents a flat auto-encoder with a light architecture and not so many parameters.

The FSRCNN model is based on the large model presented in [32], employing an structure that takes an input image in its original resolution, and up-scaling to the target resolution is performed in the final layer using a transposed convolution layer. This model is unique in that it performs all the operations in the source image resolution and performs the up-scaling in the final stage of the CNN. Hence, there is coherence in terms of training between FSRCNN and other models with similar takes on up-scaling including ESPCN [101].

The REDNet framework is based on a 10-layer hourglass-shaped encoder-decoder architecture formed by coupled convolution and transposed convolution operations, as well as application of skip connections for efficient training of the networks, with the details presented in Figure 3.2. This model is representative of several deep learning-based SR approaches in the literature, and it has coherence with models such as VDSR [63] in some aspects such as depth of the network and application of residual learning.

In the experiments described in the following sections, all the models were implemented using the TensorFlow [7] library, and two common scaling factors of 3 and 4 were considered in the validation of the methods, while for each scaling factor all networks were trained twice from scratch, once using MSE, and once using the proposed MPC. All the trainings were performed on Tesla K80 NVIDIA GPUs.

3.5.1 SRCNN Model Evaluations

The 91 images data set was used for training the SRCNN, similar to what was presented in [30]. The images were first down-sampled, and then up-scaled to the original resolution using the bi-cubic interpolation to create the input training images that are coupled with the original high resolution ground truth images. The images were partitioned into smaller patches to create more training samples. The partitioning was performed by extracting blocks of 33×33 for scaling factor of 3 and blocks of 32×32 for scaling factor of 4 with a stride of 14 for both factors. That led to around 22,000 training pairs for both scenarios. A training batch size of 128 was selected and Stochastic Gradient Descent (SGD) was used as the optimization algorithm similar to original authors' approach.

Table 3.2 summarizes the training conditions for each model including SRCNN, and also indicates the diversity of the performed evaluations with regard to the type of

TABLE 3.2: Summary of training conditions for each SR model used in the experiments.

	SRCNN	**FSRCNN**	**REDNet**
Data set	91 images	General-100 (augment)	DIV2K
Training samples ($\times 3$)	$\sim$22K	$\sim$1.8M	$\sim$2.3M
Training samples ($\times 4$)	$\sim$22K	$\sim$0.96M	$\sim$2.3M
Input-output sample size ($\times 3$)	33 - 33	11 - 33	40 - 40
Input-output sample size ($\times 4$)	32 - 32	10 - 40	40 - 40
Learning rate†	10^{-4}, 10^{-5}	10^{-3}, 10^{-4}	10^{-4}
Activation function	ReLU	PReLU	ReLU
Optimization	SGD	SGD	Adam
Total back-propagations	4,500,000	4,500,000	400,000
Batch size	128	128	128
Network parameters	$\sim$8K	$\sim$25K	$\sim$1.7M

† SRCNN and FSRCNN use two different learning rates for weights and biases.

TABLE 3.3: PSNR/SSIM analysis of bi-cubic interpolation and SRCNN and FSRCNN based on the reports in the literature.

Scaling factor	**Data**	**Bi-cubic**	**SRCNN**	**FSRCNN**
$\times 3$	Set5	30.42/0.8682	32.39/0.9033	33.16/0.9140
	Set14	27.54/0.7728	29.00/0.8145	29.43/0.8242
$\times 4$	Set5	28.44/0.8110	30.09/0.8530	30.71/0.8657
	Set14	26.00/0.7009	27.20/0.7413	27.59/0.7535

the networks, data sets, and training parameters, which helps in the validation of the proposed method.

The training for the SRCNN model was stopped after 4.5 million back-propagation iterations, which accounted for $\sim$26,470 epochs for scaling factor of 3 and $\sim$26,162 epochs for scaling factor of 4. In order to track the status of the convergence the performance of the models were tested every 100,000 iterations on the Set 5 and Set 14 images, and the PSNR values were recorded for both scaling factors and for models trained with MSE and MPC. The summary of results is demonstrated in Figure 3.3.

When training with MSE, SRCNN does not converge to the final model reported in the literature (Tabel 3.3) within 4.5 million allowed back-propagations, and much more training is expected for convergence according to the results depicted in Figure 3.3. The training was not continued due to low resources, however according to [30] some 150 million iterations is expected for SRCNN to achieve the desired model. This is coherent with the red curve in Figure 3.3, and its linear behavior, which can eventually reach the dark blue line after enough back-propagations. In spite of the poor performance of the MSE, a very swift convergence of the model can be seen when using the proposed MPC. In fact SRCNN is converged after only about 400,000 iterations using the proposed cost function, which accounts to a 97% reduction in the number of back-propagation

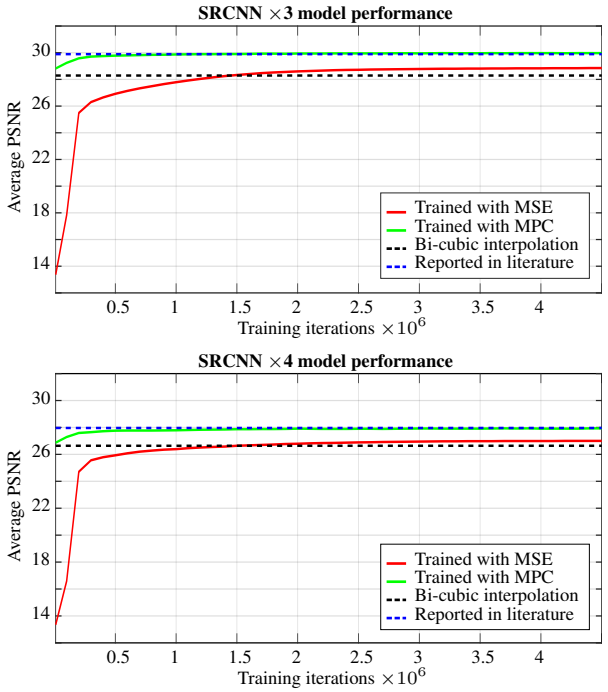


FIGURE 3.3: PSNR analysis of SRCNN models trained using MSE and MPC at every 100,000 training iterations on Set 5 and Set 14 in average.

iterations for MPC case when compared with the required number of iterations for MSE, resulting in a huge speed-up in training. Tables 3.4 and 3.5 report on the performance of the SRCNN during different stages of the training and on various data sets. The results evidence a significant deviation between the networks learned with different cost functions, implying the effectiveness of the proposed cost function in speeding up the training process for CNNs. Additionally, the final model trained with MPC (last column of Tables 3.4 and 3.5) shows a slight improvement of the quality, compared to the figures reported in the literature, which is another advantage in adopting MPC.

With regard to Figure 3.3, it is also worth mentioning that a similar behavior is expected for SSIM metric based on the results summarized in Table 3.5 and the correlation of PSNR and SSIM as depicted in Figure 2.10. It is also important to note that the first point in the horizontal axis is 10,000 and not zero, hence the better performance of the MPC. For the actual first iteration, it is expected that MPC and MSE lead to same results.

TABLE 3.4: PSNR analysis in different training stages using MSE and MPC for SRCNN on SR test sets.

Scale	Data	100K iter.		1.5M iter.		3M iter.		4.5M iter.	
		MSE	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**
×3	Set5	17.21	31.50	30.46	32.40	30.98	32.47	31.05	32.51
	Set14	18.10	28.19	27.47	29.01	27.88	29.05	27.93	29.06
	BSD100	18.50	27.69	27.17	28.13	27.45	28.17	27.50	28.18
	Urban100	16.94	24.58	23.94	25.20	24.28	25.29	24.33	25.29
×4	Set5	15.84	29.35	28.53	30.08	28.94	30.12	28.99	30.12
	Set14	16.86	26.56	25.96	27.18	26.24	27.24	26.29	27.26
	BSD100	17.26	26.33	25.97	26.62	26.14	26.66	26.17	26.67
	Urban100	15.92	23.04	22.65	23.45	22.83	23.52	22.86	23.52

TABLE 3.5: SSIM analysis in different training stages using MSE and MPC for SRCNN on SR test sets.

Scale	Data	100K iter.		1.5M iter.		3M iter.		4.5M iter.	
		MSE	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**
×3	Set5	0.5912	0.8852	0.8675	0.9026	0.8766	0.9040	0.8782	0.9043
	Set14	0.5272	0.8003	0.7739	0.8115	0.7903	0.8125	0.7929	0.8123
	BSD100	0.5047	0.7691	0.7410	0.7776	0.7574	0.7788	0.7601	0.7792
	Urban100	0.4211	0.7458	0.7105	0.7689	0.7313	0.7726	0.7348	0.7737
×4	Set5	0.5525	0.8333	0.8144	0.8519	0.8242	0.8536	0.8253	0.8545
	Set14	0.4979	0.7277	0.7073	0.7389	0.7185	0.7398	0.7209	0.7411
	BSD100	0.4775	0.6927	0.6744	0.6998	0.6844	0.7007	0.6868	0.7021
	Urban100	0.3922	0.6490	0.6243	0.6693	0.6373	0.6722	0.6402	0.6741

3.5.2 FSRCNN Model Evaluations

For FSRCNN experiments, the General-100 data set presented in [32] was used with data augmentation. Data augmentation was performed by creating multiple versions of the training set using rotation and scaling of the images. The partitioning for this data set was performed by creating low resolution blocks of 11×11 for scaling factor of 3 and 10×10 for scaling factor of 4 with a stride of 4 in both cases. It is worth noting that in the original FSRCNN approach, a pre-training is first performed using the 91 images data set to have a better initialization of the weights. However in the following experiments, the network is only trained on the General-100 images, rather than following the training approach proposed by Dong et al. [32]. The reason for this choice is to make the training process more challenging, given that pre-training using 91 images can lead to better weight and bias initializations for the larger data set, and the training problem can turn into a tuning problem. Table 3.2 summarizes the detailed parameters and conditions for FSRCNN experiments.

Similar to SRCNN experiments, the batch size was set to 128, and the SGD was used for optimization. The training for both MSE and MPC scenarios were stopped after

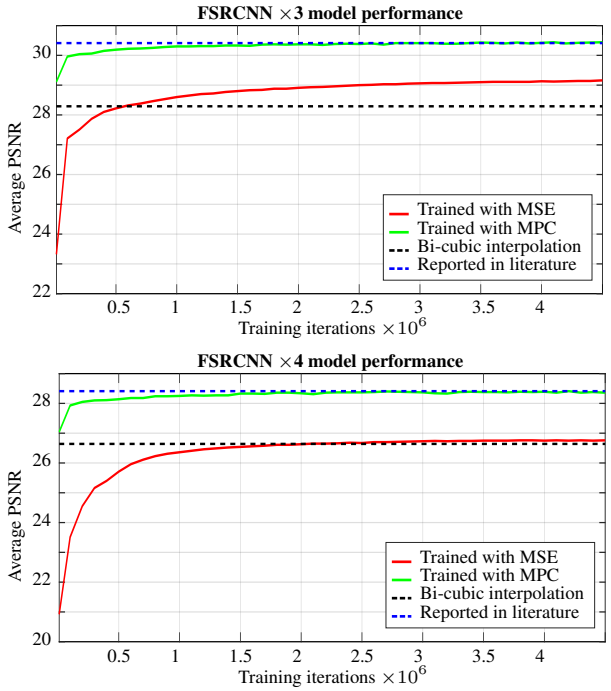


FIGURE 3.4: PSNR analysis of FSRCNN models trained using MSE and MPC at every 100,000 training iterations on Set 5 and Set 14 in average.

4.5 million iterations, and the performance of the models on Sets 5 and 14 are outlined in Figure 3.4 for every 100,000 back-propagations. Similar to the case with SRCNN, the MSE optimization is not able to train the model within the limited envisaged back-propagations. For both scaling factors, the curves in Figure 3.4 demonstrate that much more iterations are necessary for MSE minimization to result in the promised FSRCNN models, and the convergence is extremely slow. According to [32] and depending on the training approach, at least 600 million iterations are needed for convergence of FSRCNN.

Contrary to MSE, the performance of the MPC optimization is very efficient for FSRCNN, and for both scaling factors the network converges after roughly 1,800,000 iterations. This accounts for more than 99% reduction in the number of needed back-propagation iterations when compared with the original 600 million iterations of the MSE scenario. Tables 3.6 and 3.7 report on the performance of the FSRCNN during different stages of the training on various data sets, which is another way of demonstrating the effectiveness of the proposed loss function. Similar to SRCNN, the final FSRCNN model learned by MPC outperforms the literature model slightly.

TABLE 3.6: PSNR analysis in different training stages using MSE and MPC for FSRCNN on SR test sets.

Scale	Data	100K iter.		1.5M iter.		3M iter.		4.5M iter.	
		MSE	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**
×3	Set5	28.91	32.40	30.77	33.05	31.20	33.13	31.36	33.18
	Set14	25.98	28.87	27.27	29.36	27.61	29.43	27.76	29.46
	BSD100	26.20	28.10	27.30	28.38	27.48	28.43	27.54	28.43
	Urban100	22.98	25.22	24.06	25.64	24.27	25.72	24.35	25.72
×4	Set5	24.50	30.16	28.79	30.69	29.07	30.75	29.12	30.81
	Set14	23.16	27.13	25.73	27.58	25.89	27.63	25.91	27.63
	BSD100	23.56	26.67	25.99	26.86	26.11	26.92	26.11	26.90
	Urban100	20.63	23.50	22.74	23.79	22.81	23.86	22.82	23.85

TABLE 3.7: SSIM analysis in different training stages using MSE and MPC for FSRCNN on SR test sets.

Scale	Data	100K iter.		1.5M iter.		3M iter.		4.5M iter.	
		MSE	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**
×3	Set5	0.7958	0.9036	0.8754	0.9119	0.8768	0.9132	0.8788	0.9138
	Set14	0.7009	0.8131	0.7900	0.8205	0.7909	0.8221	0.7927	0.8228
	BSD100	0.6833	0.7787	0.7636	0.7859	0.7634	0.7877	0.7643	0.7883
	Urban100	0.6450	0.7716	0.7333	0.7873	0.7355	0.7907	0.7376	0.7919
×4	Set5	0.7150	0.8536	0.8101	0.8675	0.8133	0.8692	0.8064	0.8709
	Set14	0.6034	0.7419	0.7073	0.7523	0.7094	0.7542	0.7023	0.7543
	BSD100	0.5933	0.7033	0.6802	0.7109	0.6816	0.7130	0.6754	0.7129
	Urban100	0.5276	0.6741	0.6336	0.6921	0.6351	0.6959	0.6301	0.6965

3.5.3 REDNet Model Evaluations

A variant of the REDNet model [85] with ten layers is chosen as the third test scenario for the proposed training approach. Studying the behavior of REDNet is particularly important, as it is representative of a variety of state-of-the-art approaches in SR, that take advantage of deep encoder-decoder architectures with many trainable parameters, as well as incorporation of the residual learning, which can result in accurate and more importantly swift training of the network parameters. Use of residual learning, and the hourglass structure of the network can lead to a more efficient training process than the two cases studied previously, hence it is important to test such an architecture for validation of the proposed cost function.

The DIV2K data set [10] was used for training the REDNet network. The partitioning process for this set was done by creating 40×40 high resolution blocks with a stride of 30, leading to around 2,400,000 samples. Another critical aspect in REDNet training is application of the Adam optimizer [65], which in nature is a more efficient and practical optimization algorithm than the classic SGD. Adam is a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement.

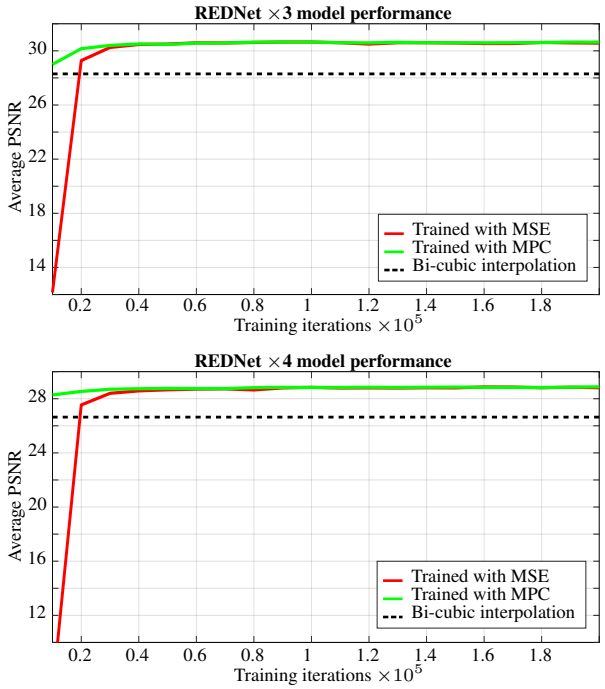


FIGURE 3.5: PSNR analysis of REDNet models trained using MSE and MPC at every 10,000 training iterations on Set 5 and Set 14 in average.

The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. Adam method is designed to combine the advantages of AdaGrad [33], which works well with sparse gradients, and RMSProp [110], which works well in on-line and non-stationary settings. Some of Adams advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradient, its step sizes are approximately bounded by the step size hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing. All these feature make Adam an interesting case study for evaluation of the proposed cost function. Hence investigation of the MPC performance with this optimizer is covered by studying the REDNet performance.

Similar to previous cases a batch size of 128 was chosen, and the network was trained for two scaling factors of 3 and 4. Given that the convergence of the REDNet is much faster than SRCNN and FSRCNN, the training for both cases of MSE and MPC was stopped after 200,000 iterations. Figure 3.5 demonstrates the performance of the model on Sets 5 and 14. As expected, MSE performs well for this architecture thanks to the Adam optimizer. However, it is interesting to see even for such a scenario, the proposed MPC is still outperforming the MSE by starting the learning at a more promising level, as well as converging faster than the MSE by 10,000 iterations, which can result in

TABLE 3.8: PSNR analysis in different training stages using MSE and MPC for REDNet on SR test sets.

Scale	Data	20K iter.		30K iter.		40K iter.		200K iter.	
		MSE	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**
×3	Set5	31.56	32.80	32.91	33.15	33.27	33.32	33.60	33.65
	Set14	28.46	29.22	29.30	29.42	29.47	29.50	29.62	29.62
	BSD100	27.81	28.33	28.38	28.48	28.52	28.56	28.68	28.71
	Urban100	24.79	25.55	25.63	25.83	25.91	25.98	26.27	26.28
×4	Set5	29.80	30.99	30.80	31.13	31.02	31.16	31.41	31.43
	Set14	26.95	27.74	27.62	27.85	27.77	27.90	27.95	28.02
	BSD100	26.51	27.01	26.93	27.07	27.05	27.10	27.17	27.17
	Urban100	23.32	24.04	23.88	24.14	24.02	24.13	24.29	24.33

TABLE 3.9: SSIM analysis in different training stages using MSE and MPC for REDNet on SR test sets.

Scale	Data	20K iter.		30K iter.		40K iter.		200K iter.	
		MSE	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**	**MSE**	**MPC**
×3	Set5	0.8820	0.9067	0.9084	0.9128	0.9142	0.9151	0.9192	0.9197
	Set14	0.7956	0.8162	0.8175	0.8218	0.8226	0.8238	0.8277	0.8279
	BSD100	0.7659	0.7833	0.7835	0.7884	0.7887	0.7903	0.7943	0.7950
	Urban100	0.7459	0.7809	0.7831	0.7926	0.7949	0.7974	0.8084	0.8088
×4	Set5	0.8357	0.8785	0.8684	0.8770	0.8745	0.8785	0.8830	0.8831
	Set14	0.7289	0.7615	0.7529	0.7602	0.7577	0.7615	0.7643	0.7652
	BSD100	0.6931	0.7189	0.7123	0.7179	0.7165	0.7189	0.7223	0.7220
	Urban100	0.6556	0.7076	0.6932	0.7076	0.7014	0.7076	0.7172	0.7186

around 30% reduction in the number of back-propagations for this model. This is yet another strong case that proves the functionality of the proposed MPC function in terms of the training efficiency when compared with MSE. To have a better insight on the performance comparison of the two cost functions, Tables 3.8 and 3.9 summarize the results of the tests after different stages of the training for various SR test sets. It is evident that even for such a robust architecture, MPC outperforms MSE in terms of training performance.

3.5.4 Error Rates during Training

Previous sections analyzed the performance of each cost function at inference, and demonstrated how the trained models behave when coping with new data as input. It is also worth monitoring the value of loss during the training for each cost function and for each of the tested models. Looking at the value of the error as the back-propagation iterations take place gives more insight on the behavior of each function, and is also a way of visualizing the stability of the training while using MPC function. In the following graphs, the scaling factor of 4 is considered, and the error rates during the first

TABLE 3.10: Complexity and runtime analysis of the back-propagations and training iterations for MSE and proposed MPC functions.

Model	Complexity metric	MSE	MPC
SRCNN	Single iteration runtime [sec]	0.108	0.112
	Total convergence time [hour]	> 140	12.23
	Total convergence iterations	$> 4{,}500{,}000$	400,000
FSRCNN	Single iteration runtime [sec]	0.088	0.090
	Total convergence time [hour]	> 110	44.91
	Total convergence iterations	$> 4{,}500{,}000$	1,800,000
REDNet	Single iteration runtime [sec]	0.405	0.427
	Total convergence time [hour]	3.39	2.37
	Total convergence iterations	30,000	20,000

200,000 training iterations for SRCNN, FSRCNN, and REDNet are depicted. Other than visibly fast convergence of the MPC, its stability in terms of the resulting error values and staying away from the lower bound of logarithm function, which is minus infinity, can be seen in Figure 3.6.

3.5.5 Evaluation of the Training Time

Although the above experiments show a promising performance for MPC in terms of reducing the number of necessary back-propagations for training SR networks, they do not reflect the amount of time being saved by using MPC instead of MSE. Moreover, given the complexity of the MPC, and the fact that its computation will naturally take more time than MSE, it is critical to compare the training time for all tested scenarios, too.

When considering the processing time of one back-propagation iteration, computation cost of the MPC is negligible in the training process, and no significant disparity can be witnessed in the runtime for each back-propagation when using MPC rather than MSE. Table 3.10 reports the runtime for each back-propagation iteration in different models using MSE and the proposed MPC function. Based on the runtime values, the overhead complexity introduced by the new cost function is insignificant, and the overall training time is extremely lower for the case of MPC. Hence the reduction in the number of training iterations is aligned and correlated with the amount of time saved during training SR networks.

FIGURE 3.6: Error rates during training for the first 200,000 back-propagations for both MSE and MPC.

3.6 Conclusions

Application of the proposed MPC function for training the SR networks improves the learning process, and speeds up the convergence of the network parameters significantly,

leading to major savings in time and resources for training CNNs. MPC is based on the logarithmic MSE and proximity function, and has high correlations with visual quality metrics widely used in SR, making it a suitable cost function for error minimization. Effectiveness of this function is illustrated through the experiments on different network architectures, and it promises an accurate and fast learning process for complex networks employing large training sets. It is also worth noting that application of MPC is not limited to SR networks, and can easily be extended to other generative models such as de-noising and de-blurring.

As mentioned earlier, this chapter was focused on the training process of SR networks. The aim was to achieve speed-ups in the training cycles of the SR models that lead to the same qualitative performance during the inference. Next chapter focuses on the inference, in that methods are introduced to enhance the quality and reduce the complexity of the SR inference. Although the training is a crucial part of every deep learning model, next chapter deals mainly with different architectural design aspects of SR models using classic MSE cost function.

The proposed cost function, MPC, is revisited again in Chapter 5, where a complex deep learning architecture is presented to perform up-scaling to ultra high resolutions using an efficient design. The training takes advantage of the MPC loss to achieve quick convergence to the desired model.

Chapter 4

Lossless Pooling Convolutional Networks for Super-Resolution

In this chapter, a novel pooling layer is proposed for deep learning applications, in particular Super-Resolution (SR). Unlike the existing pooling operations in the state-of-the-art deep learning architectures, the proposed pooling layer is lossless, i.e. no data is discarded during the compression phase. Lossless pooling can be utilized as an effective measure for complexity reduction of the SR models, as well as improving the quality of the image restoration. The functionality of the proposed method is demonstrated through several deep learning architectures for image SR, and the evaluations provide promising results in terms of both computation complexity and picture quality.

4.1 Introduction

SR in still images and videos by application of Convolutional Neural Networks (CNN) is a well-studied task in computer vision. Deep learning architectures comprising of multiple convolutional (and transposed convolutional) layers can provide high quality reconstruction and up-scaling of the visual data, and various approaches in designing such models have been proposed in recent years, with some of the pioneering contributions summarized in Chapter 2.

Deep learning-based SR models typically do not include any pooling operations in their respective architectures. Pooling layers combine the outputs of neuron clusters and map them to a single neuron in the next layers. Application of different types of pooling, such as max-pooling or average-pooling, is a common technique in numerous classification tasks for feature extraction, as well as dimension reduction of networks and improving the training performance [87, 97, 125]. Generally, pooling reduces the size of the feature maps in a selective way, resulting in complexity reduction in the future layers of CNNs. Pooling operations can result in loss of data due to the nature of operation, hence they are not a particularly good design choice for SR models that aim at reconstructing visual representations. Therefore SR models are not equipped with pooling layers, and that often leads to major computation and power issues during both training and inference phases.

In this chapter, a lossless pooling operation is proposed that can reduce the spatial size of the feature maps without discarding any data. The proposed operation can be integrated as the first layer within the SR encode-decoder architectures. The output of the lossless pooling operation can be exploited by the networks in different ways, depending on the design purposes of the SR model. In this regard, two general approaches in integration of the lossless pooling layers in deep learning-based SR models are proposed, that each can enhance the image restoration process from a different aspect.

The rest of this chapter is organized as follows. Section 4.2 presents the methodology concerning the proposed lossless pooling layer. Section 4.3 presents the two different approaches in applying the lossless pooling operations, along with sample SR architectures taking advantage of the proposed approaches. Section 4.4 reports on the experiments and systematic evaluation of the methods, followed by the conclusions in Section 4.5.

4.2 Lossless Pooling Operation

As mentioned earlier, the existing pooling operations in deep learning result in loss of data. Two very popular methods for pooling are max-pooling and average-pooling, which perform compression on a given feature map by selecting either the maximum value or the average value within a block of pixels in the feature map. Such methods are used for extracting rotational and position invariant feature by extracting the dominant feature

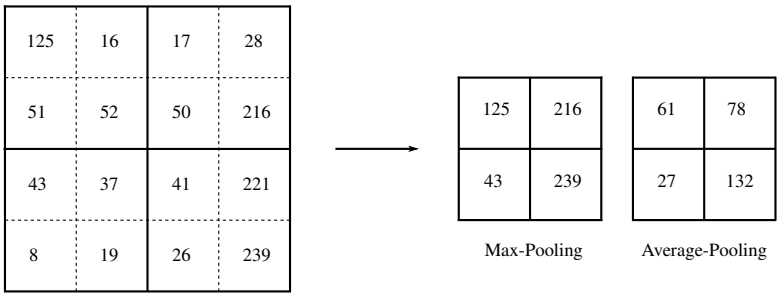


FIGURE 4.1: Examples of max-pooling and average-pooling operations with pooling window of 2×2 for a sample matrix.

values from the given region irrespective of the position of the feature value. Figure 4.1 demonstrates simple representations for these methods for a sample input matrix. Although beneficial in classification tasks, the loss of data is vivid in this example, and it is easy to foresee possible misrepresentation of the features in regression tasks when pooling is applied.

In order to avoid any loss of data, a pooling layer is proposed that down-scales a single-channel matrix to a multi-channel tensor with lower spatial resolution. This pooling mechanism with parameter r is performed by rearranging a $H \times W$ matrix $\mathbf{M}$ into a $\frac{H}{r} \times \frac{W}{r} \times r^2$ tensor $\mathbf{T}$. The operation can be considered as reverse periodic shuffling proposed in [101] and described in Section 2.3.2. Lossless pooling $\mathcal{LP}$ can be described mathematically as the following:

$$\mathbf{T}_{x,\ y,\ c} = \mathcal{LP}(\mathbf{M};\ x,\ y,\ c) = \mathbf{M}_{r \cdot x - \bmod (r^2 - c, r),\ r \cdot y - \left\lfloor \frac{r^2 - c}{r} \right\rfloor} \tag{4.1}$$

where x, y, and c represent the coordinates of a pixel in $\mathbf{T}$. Although the above description aims at lossless pooling of single-channel (grayscale) images, the concept can be

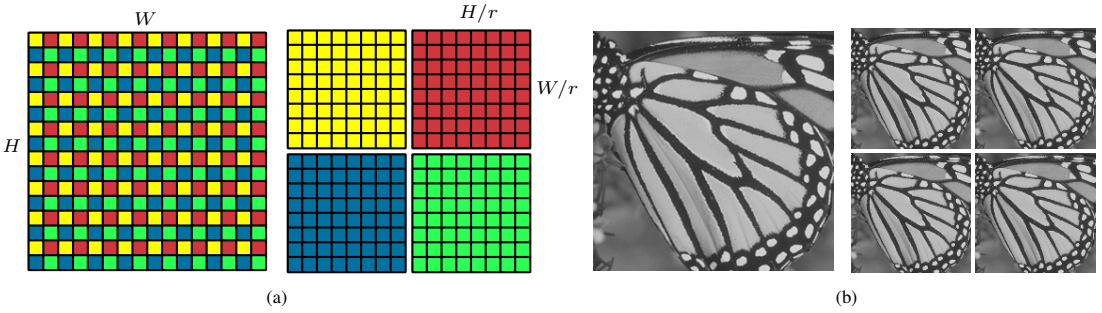


FIGURE 4.2: The lossless pooling process: (a) Input and output of the layer for $r = 2$, and (b) a sample lossless pooling on a grayscale image.

easily extended to multi-channel images. Figure 4.2 demonstrates the lossless pooling operation, along with a visual example.

Application of lossless pooling layer reduces the spatial size of the input data without losing any information, and reduces the computation cost of convolution operation in the succeeding layers by a factor of r^2. Moreover, the complexity of this pooling process is significantly lower than that of a convolution layer with a stride of r, which can also result in the down-scaling of the feature maps. This is due to the fact that lossless pooling comprises of only permutation operations, whereas convolution filter comprises of kernel operations and matrix multiplications that leads to more complex process. The next section demonstrates the use-cases for lossless pooling within SR networks, and proposes approaches for integrating this layer in encoder-decoder architectures, with further explanations on performance characteristics of the proposed process.

4.3 Advanced Super-Resolution using Lossless Pooling

Classic pooling is essentially applied for feature extraction and improving the classification performance in deep learning architectures. The dimension reduction aspect of pooling makes it impractical for application in regression and SR scenarios, due to loss of data. However, introduction of lossless pooling can handle the dimension reduction process without this loss. Dimension reduction can significantly improve the inference in SR applications, if all the feature values are retained during the process. Hence lossless pooling is a suitable candidate for performing dimension reduction in SR models.

As presented in the previous section, lossless pooling operation has two main characteristics: The output signal has a smaller spatial resolution, and the output is comprised of several replicas from the original image with high correlation. These two characteristics can be utilized in two different scenarios in SR to achieve two important goals. The main goals in improving a SR framework are reducing the complexity of the approach by decreasing the inference computation time, as well as improving the visual quality of the reconstructed images. The characteristics of the lossless pooling operation can address both goals.

The proposed approach in applying the lossless pooling in an SR framework is to integrate it as the first layer of the deep learning architecture, hence acting as a pre-processing step in the SR pipeline. Given an image as the input to the SR model, the lossless pooling operation can provide a set of lower resolution self-replicas of the input image, and depending on how the fusion of the self-replicas is performed in the CNN, various network architectures can be devised.

In this regard, two general approaches are proposed for integration of the lossless pooling in deep learning-based SR models. Both approaches integrate the lossless pooling as the first layer of a deep encoder-decoder architecture, however the first approach is designed to reduce the complexity of the SR inference, particularly for high resolution content, whereas the second approach mainly focuses on improving the image quality by smart fusion of the input self-replicas in the SR network. The two approaches are discussed in details in the following.

4.3.1 Fast Lossless Pooling Networks for SR

A typical approach in designing SR networks is up-scaling the input image to the target resolution using bi-cubic interpolation, hence the input and output images in the CNN have similar spatial resolution. This means the convolutional operations start at the highest resolution, resulting in an increase in the number of computations in the network. Shi et al. [101] and Dong et al. [32] addressed this issue and devised a solution to reduce the complexity of the SR frameworks. In their approaches, the input image remains as it is and is fed directly to the SR network, therefore all the convolutional operations happen in the input native low resolution space. The actual up-scaling of the image takes place in the final layer of the CNN using a sub-pixel scaling [101] or a transposed convolution layer [32]. These approaches reduce the complexity of the inference significantly and were also applied more recently in several still image SR contributions [72, 77, 113]. However, such models share a critical deficiency, which is the inconsistency of the network structure for different scaling factors. These models require different structures for each scaling factor, and that leads to dependence of the complexity to the scaling factor, as well as the input image resolution.

More importantly, the above low-complexity approaches fall behind the state-of-the-art models in terms of the image quality, and cannot achieve accurate image restoration.

The best existing approaches in still image SR require deep network architectures with many convolutional layers. Such models although very promising in terms of quality, cannot be utilized for up-scaling to very high resolutions, e.g. for broadcasting or camera applications, and they often function very slowly on CPUs. This has been the inspiration to propose a network architecture that can still perform reasonably fast on CPUs for very high resolution SR, while providing high quality image enhancement. The core idea is to down-scale the input image to a lower resolution without losing information using the proposed pooling mechanism, hence the enhancement process is performed on a smaller spatial space, and the processing time is reduced. The enhanced data is then up-scaled back to the original resolution using sub-pixel convolution layer and periodic shuffling.

In order to devise the low complexity SR architecture, an SR framework is required which improves the quality of an image already up-sampled to the target resolution (e.g. by bi-cubic interpolation). An hourglass-shaped CNN with encoder-decoder structure inspired by the REDNet network proposed in [85] is deployed. This network forms the basis of the restoration framework.

The lossless pooling layer is integrated as an initial layer to the REDNet CNN to reduce the complexity. The up-scaled single-channel input image is reshaped into a four-channel image with smaller spatial resolution, and the outcome is fed to the first convolutional layer. In order to reconstruct the high resolution image at the end of the process and achieve the target resolution, a sub-pixel up-scaling operation, as defined in [101], is applied. For the lossless pooling parameter, $r = 2$ is selected, and consequently the periodic shuffling at the other end of the network will have the same parameter. It is worth noting that selection of r is irrespective of the targeted SR scaling factor, and can be chosen completely independent of the enhancement network. Moreover, the presented architecture can be easily tuned to different scaling factors and different levels of enhancement. Additionally, the model can be trained simultaneously for several scaling factors at the same time, unlike other low complexity deep learning-based SR models, such as [101] and [32], that require separate models for different scaling factors. The end-to-end model is called Fast Lossless Pooling Network (FLPN) and can be formulated as the following:

$$\mathbf{X}^* = \theta^F(\mathbf{Y}, r) \tag{4.2}$$

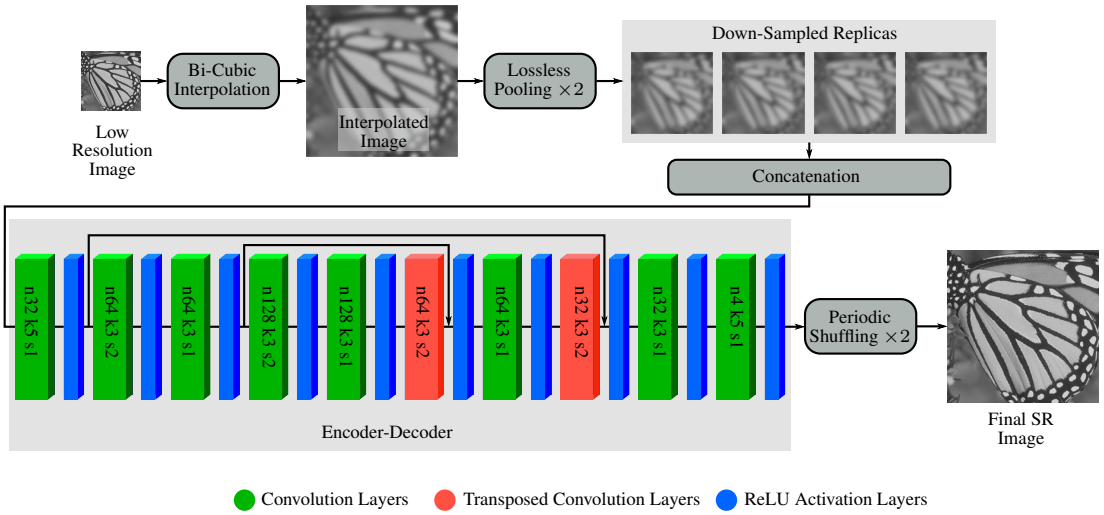


FIGURE 4.3: The architecture of the proposed FLPN for fast SR using lossless pooling. The parameters n, k, and s specify the number of filters, kernel size, and stride value for every layer.

where $\mathbf{Y}$ represents the low quality input image, up-scaled to the target resolution using an interpolation operation, $\mathbf{X}^*$ represents the high quality version of the input image created by the network output, θ^F represents all the CNN parameters including the kernels and biases within each layer, and r represents the scaling parameters for lossless pooling and periodic shuffling layers.

Figure 4.3 depicts the architecture of the proposed model for FLPN. As illustrated, the network takes advantage of coupled convolutional and transposed convolutional layers, along with skip connections, that can lead to swift and accurate training of the CNN. Moreover, application of convolution layers with strides of higher than one leads to further complexity reduction along the enhancement work-flow. All the convolution and transposed convolution layers are accompanied by ReLU layers, and the kernel sizes are specified in Figure 4.3. It is important to note that the presented SR approach is basically a complexity-reduced version of the REDNet model, which is essentially the core enhancement part of the model. In principle, REDNet is the proposed model without the lossless pooling and the periodic shuffling steps. The REDNet model in this architecture is similar to the REDNet architecture presented in Chapter 3, except with half number of filters in each layer. In theory, the REDNet complexity is r^2 times the complexity of the proposed architecture. Apart from the last convolution layer, the rest of the layers remain unchanged in a typical REDNet structure. However the size of the inputs for each of the layers in an original REDNet platform will be r^2 times of the case presented as FLPN, leading to higher complexity in computations.

The presented FLPN architecture is a sample architecture based on REDNet taking advantage of lossless pooling, however almost any encoder-decoder architecture can be applied as the baseline. More importantly, even the SR models that perform the scaling process within the network such as ESPCN [101], FSRCNN [32], and EDSR [77] can benefit from application of lossless pooling based on the described integration scheme. In this chapter, however, REDNet was chosen as the baseline in order to have a unified architecture for all scaling factors and a good reference for performance evaluations.

4.3.2 Accurate Lossless Pooling Networks for SR

While the focus of this chapter is single image SR, the inspiration for devising the next deep learning architecture comes from multi-frame SR concepts. In multi-frame SR [23, 61, 84, 109], unlike the typical still image SR problem, several input frames that are highly correlated contribute in generation of one high quality up-sampled target image, hence a better objective and subjective quality can be expected when compared to the single image SR. Based on this approach, an Accurate Lossless Pooling Network (ALPN) architecture is devised that utilizes the multi-frame concept within the still image frameworks for an enhanced SR experience, using artificially generated self-replicas of the input image created by lossless pooling operation.

Self-Replicas Fusion

In multi-frame SR, application of multiple input frames for creating a high quality up-sampled target frame is well studied, and Caballero et al. [23] and Kapperler et al. [61]

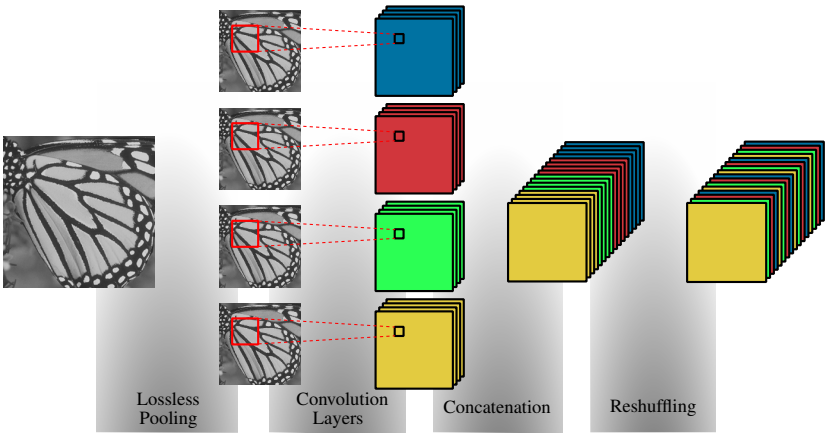


FIGURE 4.4: The process of fusing the self-replicas in CNNs by concatenation and reshuffling of the feature maps.

present different ways of fusing the inputs within the CNN architectures to obtain a single-image output from a multi-frame input, while fully exploiting the correlations between the input frames. A similar approach is adopted here, namely early fusion, for coping with the down-sampled replicas created by the lossless pooling layer.

If an input frame of size $H \times W$ is fed to the lossless pooling layer with parameter r, r^2 down-sampled replicas are generated, and each replica is fed to a convolutional layer with n filters. The resulting output is an ensemble of r^2 feature maps, each with the size $\frac{H}{r} \times \frac{W}{r} \times n$. The feature maps can then be concatenated and create one set of feature maps $\mathbf{F}$ with the size $\frac{H}{r} \times \frac{W}{r} \times nr^2$. This is similar to the early fusion concept introduced in [23, 61], also demonstrated in Figure 4.4 for the case of $r = 2$ and $n = 4$.

In addition to the concatenation of the feature maps, which is a widely used approach in multi-frame processing, reshuffling of the feature maps is performed in order to create a better mix of the features produced by different replicas. The reshuffling $\mathcal{RS}$ is performed by rearranging the order of the features in the depth dimension as the following:

$$\mathbf{F}^*_{x,\ y,\ c} = \mathcal{RS}(\mathbf{F};\ x,\ y,\ c) = \mathbf{F}_{x,\ y,\ \left\lceil \frac{c}{r^2} \right\rceil + n \cdot \bmod (c-1, r^2)} \tag{4.3}$$

where the reshuffled output of the process, $\mathbf{F}^*$, is depicted in Figure 4.4, while x, y, and c representing the coordinates of a pixel in $\mathbf{F}^*$. This output can be treated as a set of normal feature maps and be further processed in a CNN with different layers. Similar to the lossless pooling process, the concatenation and reshuffling processes can also be easily extended to multi-channel images.

ALPN: Basic Architecture

The proposed CNN architecture for still image SR can be operational in two modes. The first mode is a basic version of the model, namely ALPN, that only takes the down-sampled replicas as the key inputs for SR, and ignores the original low quality input image generated by bi-cubic interpolation. This model is depicted by solid lines and connections in Figure 4.5. The output to this model is denoted as *SR Image A* in the figure. As mentioned earlier, an input grayscale image of size $\frac{H}{s} \times \frac{W}{s}$ is interpolated to the target resolution of $H \times W$ using bi-cubic filtering. The first step after the bi-cubic interpolation is the lossless pooling with $r = 2$, which results in four down-sampled replicas, each with the size of $\frac{H}{2} \times \frac{W}{2}$. Each of the replicas goes through separate

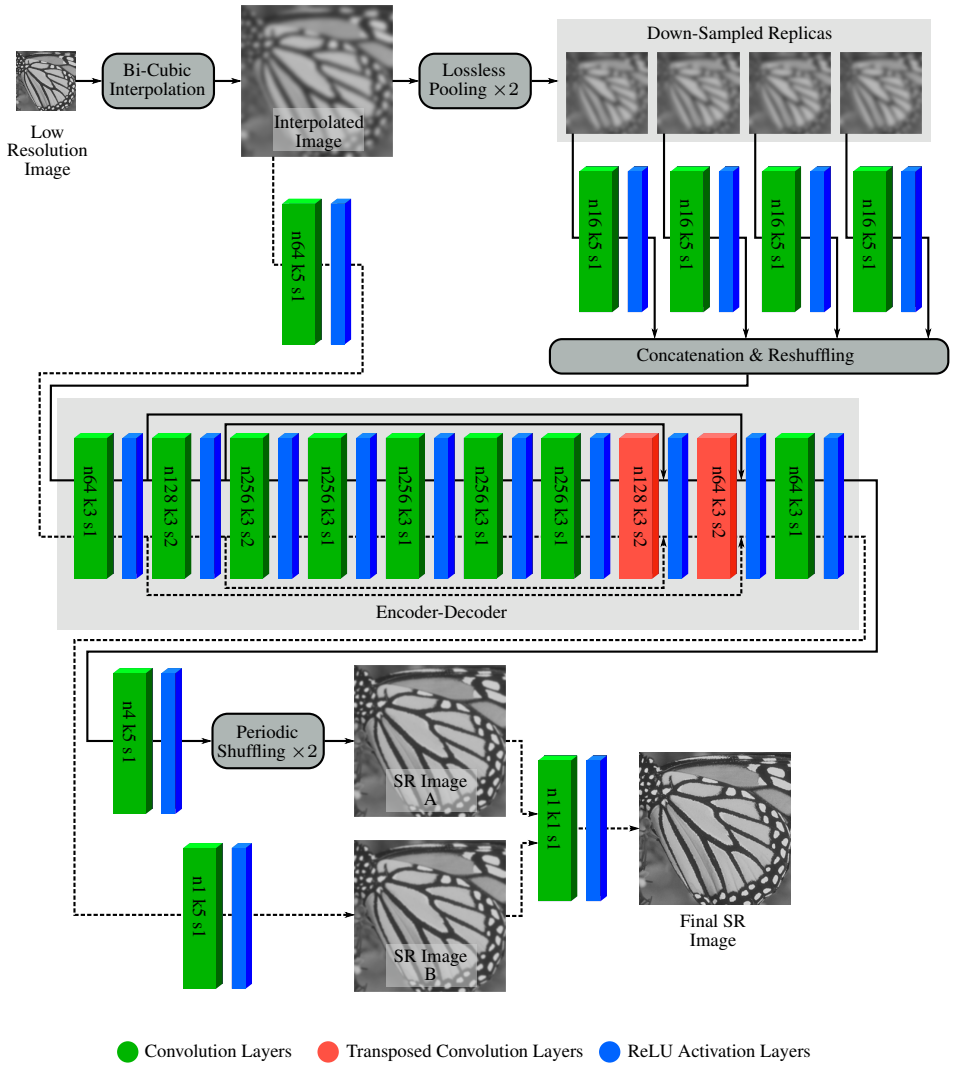


FIGURE 4.5: The architecture of the proposed accurate SR model using lossless pooling. The solid connections depict the ALPN model. Inclusion of the dashed connections in the model results in the ALPN$^+$. The parameters n, k, and s specify the number of filters, kernel size, and stride value for every layer.

convolutional layers with 16 filters, resulting in four tensors of size $\frac{H}{2} \times \frac{W}{2} \times 16$. The four tensors are concatenated and reshuffled to create a feature map of size $\frac{H}{2} \times \frac{W}{2} \times 64$.

The resulting feature map is then fed to a typical encoder-decoder architecture consisting of 10 layers with the kernel specifications presented in Figure 4.5. The encoder-decoder structure, inspired by REDNet architecture, contains coupled convolution and transposed convolution layers with stride of 2, that are also connected using skip connections. The output of this stage is a tensor with a similar size as the input feature map to the encoder-decoder network. The generated feature map is then fed to one last convolution layer resulting in a tensor of size $\frac{H}{2} \times \frac{W}{2} \times 4$. A periodic shuffling operation is then used to up-sample the tensor to the target resolution and create an image of the target size

$H \times W$. This image is denoted as *SR Image A*, and is a high quality reconstruction of the input image created by the down-sampled replicas as the only input to the model. The end-to-end model θ^A can be formulated as the following:

$$\mathbf{X}^* = \theta^A(\mathbf{Y}, r) \tag{4.4}$$

ALPN$^+$: Full Architecture

In addition to the down-sampled replicas created by the lossless pooling, the original bi-cubic version of the image can also be incorporated in the network, and provide more information for creating a higher quality SR image. This process is depicted using the dashed lines and connections in Figure 4.5, complementing the model described previously. The first step after the bi-cubic filtering is a convolution layer with 64 filters that results in a feature map of size $H \times W \times 64$. This feature map is then fed to the same encoder-decoder network described earlier with the same weights. Weight sharing is applied for this section of the model in order to avoid extra complexity and over-fitting. The output of the encoder-decoder network will be a feature map of size $H \times W \times 64$, which is then fed to a single kernel convolution layer, that results in a high quality SR image, denoted as *SR Image B* in the figure.

The two created SR images, *SR Image A* and *SR Image B*, are then combined using a single kernel convolution layer with the kernel size of 1×1 to create the *Final SR Image*. The final convolution layer operates as an averaging mechanism to mix the two created SR images. It is worth noting, that the encoder-decoder architecture chosen for both ALPN schemes are slightly different than the one presented for FLPN, although the general structure and the concept remain the same. The end-to-end model θ^{A+} can be formulated as the following:

$$\mathbf{X}^* = \theta^{A+}(\mathbf{Y}, r) \tag{4.5}$$

Similar to FLPN, although a REDNet-based model was taken as the core encoder-decoder engine for the proposed ALPN architecture, any other SR model built upon the auto-encoder concepts can be applied in conjunction with the lossless pooling and the proposed self-replicas fusion mechanism for providing accurate single image SR.

4.3.3 Training

Training the proposed SR models is performed by solving an optimization problem to minimize the error between the ground truth and the high resolution output images. For all cases, the training ground truth are a set of high resolution image samples $\mathbf{X}$, and the network outputs are the high quality high resolution images $\mathbf{X}^*$ generated by the models

The Mean Squared Error (MSE), defined as the following, is employed as the cost function for the training process.

$$J_{MSE}(\theta^m; \mathbf{X}, \mathbf{Y}) = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} (\mathbf{X}_{i,j} - \theta^m(\mathbf{Y})_{i,j})^2 \tag{4.6}$$

with W and H denoting the dimension of output images, and m representing the model choice. The training input samples are created by down-sampling the high resolution samples, and up-scaling them back to the original resolution by bi-cubic interpolation. The scaling factor can be fixed to focus on training a particular scaling, or alternatively can include several values to cover a wide range of scaling.

It is worth noting that the Modified Proximity-based Cost (MPC) function presented in (3.7), could have been used in this chapter instead of MSE. However, since all the experiments are based on the comparison with state-of-the-art models that rely on MSE for their training, it was decided to adopt MSE for the proposed architectures in this chapter, too, to have a fair comparison with other methods and focus only on impact of the network architectures on performance.

4.4 Experiments

The focus of the experiments were on the SR with scaling factors of 3 and 4, which are challenging factors in image up-sampling and common for benchmarking. The DIV2K data set [10] was used, which comprises 800 high quality images as the training set. The images were partitioned into 96×96 samples with a stride of 80, which led to about 325,000 training sample pairs. The r parameter was also selected to be 2 for all the

architectures. Adam optimizer [65] was employed for training the models with a learning rate of 0.0001, β_1 of 0.9, β_2 of 0.999, ϵ of 10^{-8} and training batch size of 128. For FLPN, the learning rate remained constant during the training to resemble the training scenario of the REDNet, which is the main baseline for that architecture. For ALPN models, however, the learning rate was reduced by a factor 0.8 every 15 epochs. Given that the ALPN has more parameters, an adaptive learning rate approach was adopted for training. The proposed models, along with the existing state-of-the-art approaches, were implemented using TensorFlow [7] library. The models were compared with well-known deep learning-based image SR models including SRCNN [30], ESPCN [101], FSRCNN [32], and of course REDNet [85]. All the models were implemented and trained according to the provided information in the literature. Since the selection of the data sets is not fully standardized in SR, some of the existing models have been trained and tested using data that is not publicly available. Therefore, all the models were trained from scratch to be benchmarked against a unified test set for fair comparisons. All the trainings were performed on Tesla K80 NVIDIA GPUs, and all the tests were performed on a machine with a generic Intel Core i7-6700 CPU with a 3.40GHz clock and 16GB RAM and a GeForce GTX 1070 GPU.

4.4.1 Quantitative Evaluation of FLPN

Given the nature of FLPN, and the fact that the main purpose in designing the architecture is to reduce the complexity of the baseline REDNet approach, two high resolution data sets were used for testing the performance of the proposed model. The reason for selecting high resolution data sets for FLPN evaluation is that the functionality of the method is magnified when applied on high resolution content, hence the standard SR test sets are not used in this section. The validation set of DIV2K, comprising 100 images with 2K ($\sim 2048 \times 1080$) resolution, along with the the Ultra Eye Tracking (UET) data set [89], comprising 41 high quality images with 3840×2160 resolution were used for testing. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) index were the quality metrics, and the runtime on GPU and CPU was the complexity metric for the evaluation. The aim is to achieve the same quality as the REDNet model with a significantly lower computation time for high resolution image restoration.

TABLE 4.1: Quality and complexity analysis of the FLPN and state-of-the-art approaches for the scaling factor of 3.

		Bicubic	**SRCNN**	**ESPCN**	**FSRCNN**	**REDNet**	**FLPN**
DIV2K	PSNR	29.60	30.92	31.10	31.35	31.62	31.57
	SSIM	0.8285	0.8592	0.8625	0.8677	0.8733	0.8733
	GPU time	-	0.71	1.88	0.80	1.84	4.02
	CPU time	-	2.89	0.78	0.63	5.35	2.68
UET	PSNR	32.75	33.97	34.15	34.34	34.55	34.53
	SSIM	0.8760	0.8976	0.9000	0.9033	0.9069	0.9074
	GPU time	-	1.08	4.04	1.21	4.13	9.80
	CPU time	-	130.36	2.16	1.75	156.22	7.59

TABLE 4.2: Quality and complexity analysis of the FLPN and state-of-the-art approaches for the scaling factor of 4.

		Bicubic	**SRCNN**	**ESPCN**	**FSRCNN**	**REDNet**	**FLPN**
DIV2K	PSNR	28.07	29.10	29.26	29.47	29.78	29.76
	SSIM	0.7724	0.8015	0.8060	0.8122	0.8201	0.8205
	GPU time	-	0.71	1.32	0.73	1.83	4.02
	CPU time	-	2.93	0.49	0.37	5.32	2.69
UET	PSNR	30.97	31.90	32.06	32.17	32.46	32.44
	SSIM	0.8293	0.8494	0.8529	0.8562	0.8617	0.8621
	GPU time	-	1.08	2.63	1.05	4.14	9.78
	CPU time	-	121.01	1.32	1.01	151.93	7.61

Tables 4.1 and 4.2 summarize the test results for the presented model, in comparison with state-of-the-art methods. According to the results, the proposed FLPN model can perform as good as the REDNet, which is the baseline approach for this work, promising a high quality enhancement for images.

As the main focus of the FLPN architecture is to achieve fast image restoration, it is important to analyze the processing time of different approaches, too. The processing time for the proposed method shows a massive reduction of the complexity in comparison with REDNet (96%) for UHD content, while providing the same image quality.

This promises a very efficient performance on CPU devices, as well as easy integration in devices with moderate processing power, such as smart phones and tablets, as well as cloud systems. When compared with ESPCN and FSRCNN, although FLPN is still slower, the image quality is significantly better using the proposed method. Moreover, the method has a consistent structure and complexity regardless of the scaling factor, whereas in ESPCN and FSRCNN, the complexity of the structure is dependent on the scaling factor, which in both cases scaling up to a target resolution becomes slower, when the scaling factor is lower. It is also worth noting that selecting a higher r parameter in the lossless pooling approach will result in further complexity reduction of the SR process, although affecting the quality.

4.4.2 Qualitative Evaluation of FLPN

It is evident that the application of the lossless pooling layer in the deep learning architectures using the FLPN approach results in major speed-ups in the inference, thus reducing the complexity of the still image SR significantly. In this regard, it is interesting to find realistic applications for such approach, so the presented idea can be put in real test.

One practical way of showing the effectiveness of this approach is by exploiting the concept for creating an enhancement mechanism for digital zoom in mobile phone cameras. An important and appealing feature for users in digital cameras is the zooming functionality that can enable them to get close-up views of the scenes and provide a photographic degree of freedom. Zooming can be categorized into two different classes of optical zoom and digital zoom based on the adopted technology. Optical zoom is performed by changing the focal length of a zoom lens, and is widely available in professional cameras and can promise a consistent quality of image from different angles. Digital zoom, on the other hand, handles the zooming operation by selecting (cropping) a portion of the pixels in the camera lens and performing an interpolation to reach the desired image size.

Digital zoom is the method of choice in smart phones due to the existing limitations in the hardware, and as experienced by any naïve user, it cannot guarantee a very high quality imaging when compared with the optical zoom. Digital zoom is essentially performed by spatial up-sampling of a cropped image in mobile phone cameras. Hence SR can be exploited as a tool for enhancing this feature in smart phones. The application of SR, thus, can be considered as an enhancement stage on the already zoomed-in and up-sampled image. Given that the FLPN performs a bi-cubic interpolation as a pre-processing step, the built-in interpolation process in the cameras can also be considered as a pre-processing step prior to the SR-based enhancement stage.

Today's mobile cameras provide very high resolution images, and performing deep learning-based SR for such resolutions is computationally expensive, and requires major processing power. However, it was demonstrated previously that the main strength of the FLPN is its efficiency when running on CPU platforms. Therefore, it can easily be

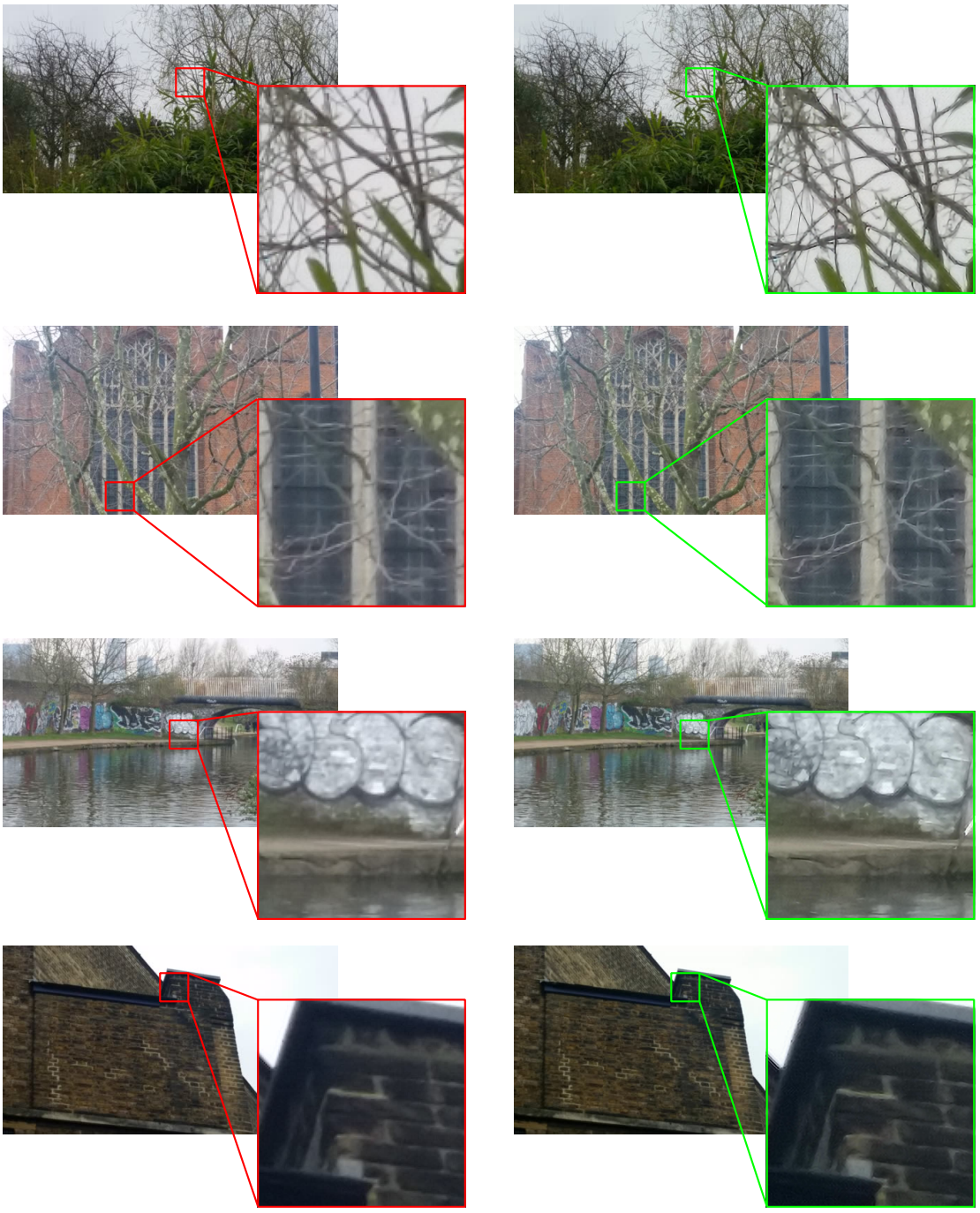


FIGURE 4.6: Quality enhancement of digital zoom: The original captured images using Samsung Galaxy S5 with $\times 4$ digital zoom (left), and the enhanced images (right).

adopted for enhancing digital zoom in mobile phone cameras, that rely heavily on CPU for any processing due to very limited graphical resources.

The application of the previously trained FLPN model for improving the quality of digital zoom was tested with real photos taken by a mobile phone in maximum zoom-in mode. A Samsung Galaxy S5 phone was used, that has a 16-mega-pixel camera,

and allows a $\times 4$ digital zoom. The camera was tested with maximum zoom, which results in photos that are interpolated to the 5312×2988 resolution. Application of the proposed enhancement approach resulted in clear visual improvements of the photos in a reasonable time (less than 10 seconds), some of which are presented in Fig. 4.6. In these examples only the luma signal is enhanced by the FLPN model.

4.4.3 Quantitative Evaluations of ALPN

The main goal of the ALPN architecture is to enhance the quality of the baseline and improve the performance of the restoration process in terms of visual fidelity and consistency of images. To compare the model with the existing solutions, the common test sets for evaluation of SR models were chosen, that include Set 5 [17], Set 14 [136], BSD 100 [86], and Urban 100 [48] data sets, which are all widely used in research community.

The same state-of-the-art approaches used for evaluation of FLPN (Section 4.4.1) were used in the following experiments, and similar testing conditions were applied for scaling factors of 3 and 4. Tables 4.3 and 4.4 summarize the performance results of the presented model, in comparison with state-of-the-art methods. According to the results, the proposed approach can outperform the baseline models in all data sets, promising a high quality enhancement for still images.

The basic ALPN model performs as good as the best existing model, REDNet, and in some cases outperforms it slightly. The strength of this model, however, is in the lower computation cost on GPU, which is due to using lossless pooling, that results in downscaling of the input, and consequently performing all the convolutional processes in a lower resolution than the native target resolution. The full ALPN$^+$ model, on the other hand, shows a solid outperformance on all data sets and provides major improvements in PSNR and SSIM results. The computation cost is however higher than the existing models due to the structure of the model, and incorporating multiple input signals in the approach.

The subjective quality of the images enhanced by ALPN$^+$ were also examined on the test content, as objective metrics cannot always grasp the intricacies detected by human eyes. Application of the proposed approach resulted in clear visual improvements in the reconstructed high resolution images, some of which are presented in Figures 4.7-4.10.

TABLE 4.3: Quality and complexity analysis of the ALPN and ALPN$^+$ and state-of-the-art approaches for the scaling factor of 3.

		Bicubic	**SRCNN**	**ESPCN**	**FSRCNN**	**REDNet**	**ALPN**	**ALPN$^+$**
Set 5	PSNR	30.42	32.52	32.81	33.20	33.65	33.74	34.01
	SSIM	0.8687	0.9048	0.9083	0.9144	0.9197	0.9219	0.9253
	GPU time	-	0.50	0.58	0.52	0.65	0.55	0.82
	CPU time	-	0.12	0.06	0.04	0.28	0.29	1.14
Set 14	PSNR	27.53	29.06	29.25	29.47	29.64	29.65	29.82
	SSIM	0.7725	0.8121	0.8173	0.8226	0.8273	0.8279	0.8330
	GPU time	-	0.53	0.66	0.56	0.73	0.59	1.01
	CPU time	-	0.24	0.10	0.06	0.54	0.54	2.24
BSD 100	PSNR	27.11	28.17	28.27	28.44	28.63	28.61	28.79
	SSIM	0.7367	0.7787	0.7830	0.7879	0.7932	0.7941	0.7988
	GPU time	-	0.51	0.63	0.55	0.71	0.66	0.99
	CPU time	-	0.14	0.08	0.04	0.39	0.40	1.61
Urban 100	PSNR	23.94	25.33	25.49	25.79	26.25	26.28	26.49
	SSIM	0.7087	0.7728	0.7779	0.7910	0.8071	0.8105	0.8163
	GPU time	-	0.52	0.65	0.55	0.66	0.58	1.08
	CPU time	-	0.20	0.09	0.05	0.40	0.42	1.53

TABLE 4.4: Quality and complexity analysis of the ALPN and ALPN$^+$ and state-of-the-art approaches for the scaling factor of 4.

		Bicubic	**SRCNN**	**ESPCN**	**FSRCNN**	**REDNet**	**ALPN**	**ALPN$^+$**
Set 5	PSNR	28.44	30.09	30.41	30.58	31.38	31.44	31.71
	SSIM	0.8110	0.8520	0.8590	0.8658	0.8820	0.8833	0.8872
	GPU time	-	0.57	0.45	0.55	0.61	0.54	0.83
	CPU time	-	0.11	0.02	0.02	0.27	0.27	1.17
Set 14	PSNR	26.00	27.18	27.37	27.52	27.98	27.98	28.12
	SSIM	0.7009	0.7385	0.7457	0.7506	0.7636	0.7643	0.7686
	GPU time	-	0.60	0.46	0.58	0.70	0.56	1.03
	CPU time	-	0.22	0.03	0.03	0.55	0.53	2.39
BSD 100	PSNR	25.89	26.64	26.77	26.85	27.16	27.14	27.26
	SSIM	0.6651	0.6994	0.7073	0.7100	0.7207	0.7215	0.7253
	GPU time	-	0.57	0.44	0.57	0.68	0.59	0.96
	CPU time	-	0.15	0.02	0.02	0.36	0.36	1.57
Urban 100	PSNR	22.58	23.56	23.67	23.85	24.29	24.21	24.48
	SSIM	0.6155	0.6741	0.6800	0.6965	0.7147	0.7172	0.7277
	GPU time	-	0.53	0.61	0.52	0.67	0.54	1.05
	CPU time	-	0.20	0.07	0.04	0.39	0.46	1.54

In these examples only the luma signal is up-sampled by the proposed SR approach, and the color components are scaled using bi-cubic interpolation.

4.4.4 Qualitative Evaluations of ALPN

One of the most challenging scenarios in SR is coping with the low resolution images with major aliasing distortions. The *Barbara* image from Set 14 of the test sets is an extreme example for this case. Up-scaling the down-sampled version of this image can result in magnification of those aliasing artifacts in most cases. However, as depicted in Figure 4.7, ALPN$^+$ does a good job in restoring some of the content that is highly distorted by

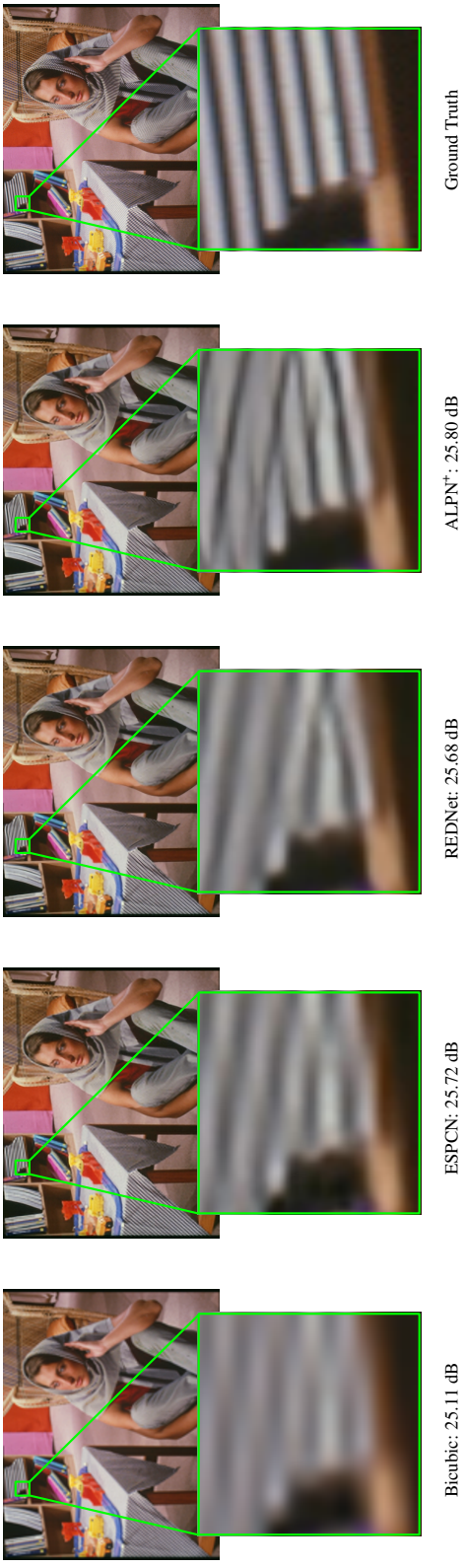


FIGURE 4.7: Up-scaling *Barbara* from Set 14 with a factor of 4 (180×144 to 720×576).

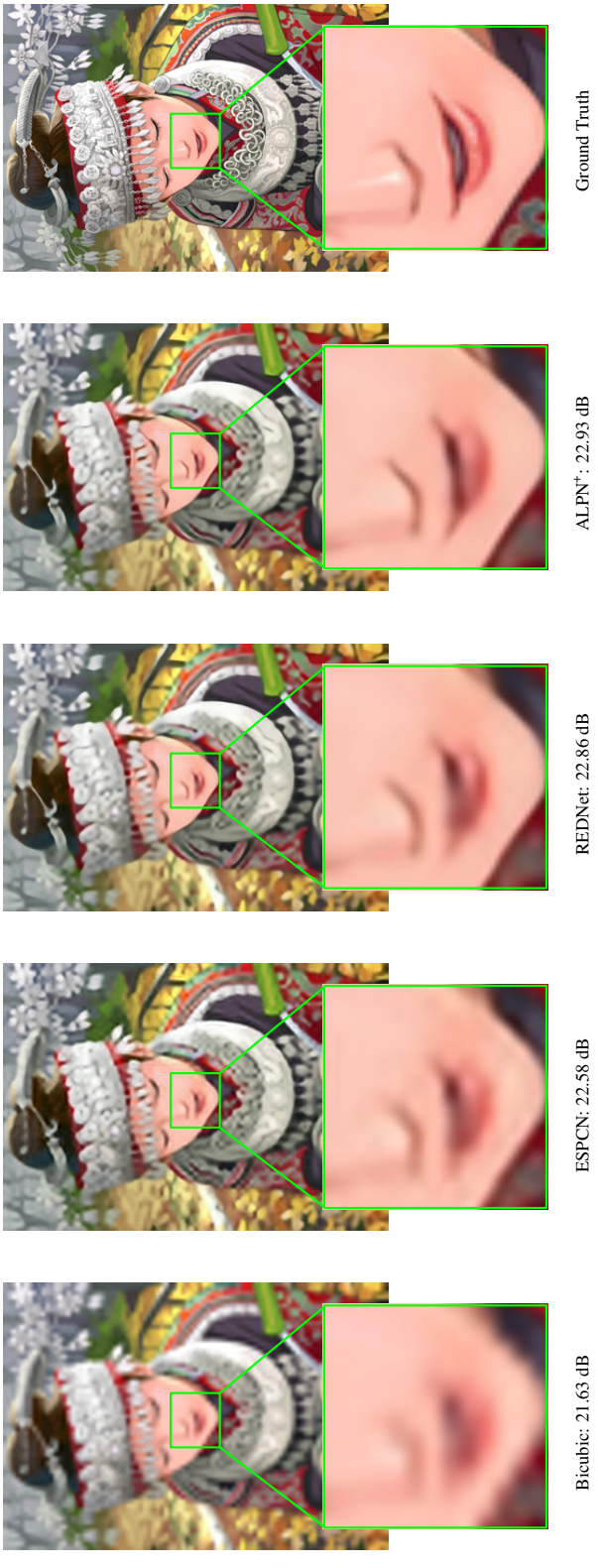
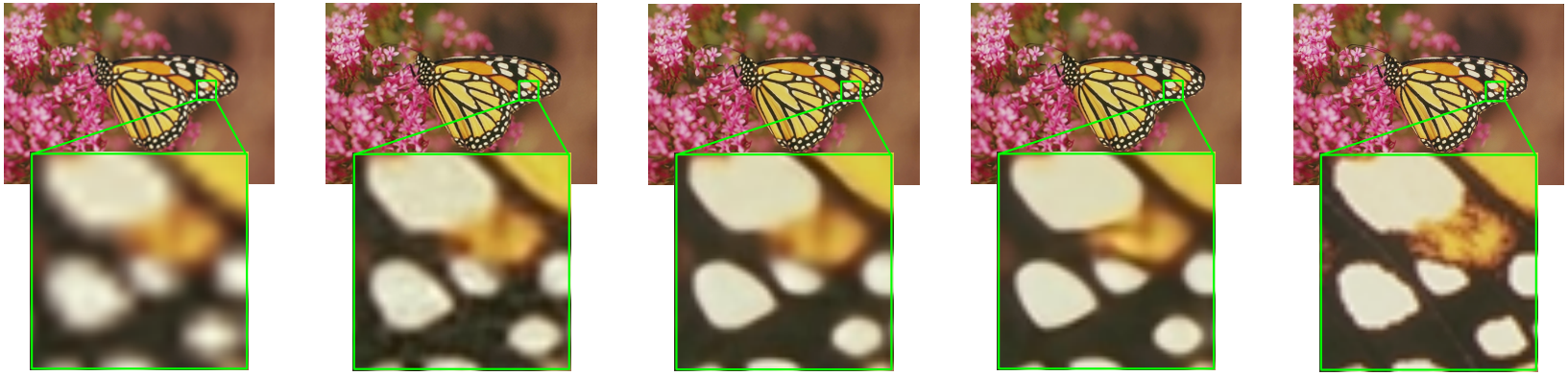
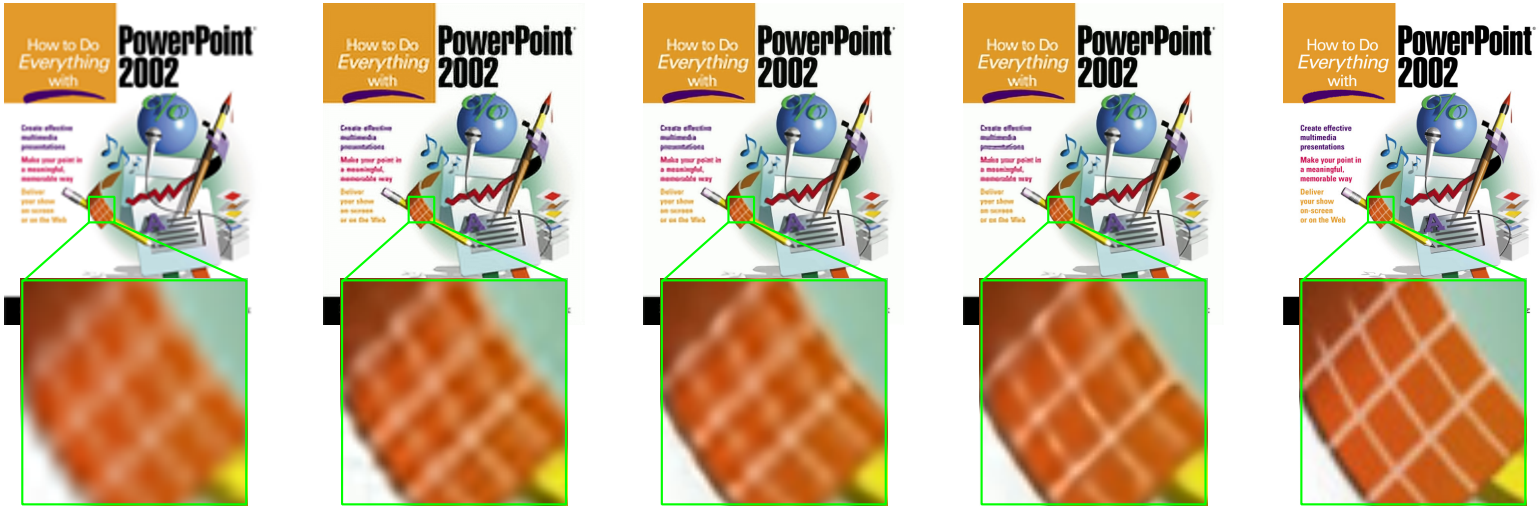


FIGURE 4.8: Up-scaling *Comic* from Set 14 with a factor of 4 (62×92 to 248×360).



Bicubic: 27.32 dB | ESPCN: 29.79 dB | REDNet: 31.27 dB | ALPN^{+}: 31.82 dB | Ground Truth

FIGURE 4.9: Up-scaling *Monarch* from Set 14 with a factor of 4 (192×128 to 768×512).



Bicubic: 21.76 dB | ESPCN: 23.74 dB | REDNet: 24.91 dB | ALPN^{+}: 25.14 dB | Ground Truth

FIGURE 4.10: Up-scaling *PPT3* from Set 14 with a factor of 4 (132×164 to 528×656).

the down-sampling process, and is outperforming the other SR approaches. Another example is in the *PPT3* image, depicted in Figure 4.10, in which the reconstruction of a distorted patterned texture is shown, and ALPN$^+$ provides a better prediction of the original image compared to the state-of-the-art models.

4.5 Conclusions

In this chapter a lossless pooling mechanism was introduced that can be integrated in state-of-the-art deep learning architectures for still image SR, and provides improvements in the performance of the SR models either in terms of complexity and computation time, or picture quality.

Two main approaches in deploying the lossless pooling layer were studied. The first approach was focused on achieving major speed-ups in network inference and reducing the complexity of the SR process for still images, and experiments promised more than 90% time saving in comparison with baseline for UHD up-scaling using the proposed model. The second approach was focused on achieving higher picture quality and more accurate reconstruction by integrating the lossless pooling and the highly correlated self-replicas in the CNN architectures, and the experiments showed significant perceptual quality improvement for the proposed architecture when compared with baseline model.

This chapter addressed the second main contribution in this thesis, focusing on systematic improvement of the performance in deep learning-based SR models both in terms of computation complexity and picture quality. Next chapter dives into applications of SR, and serves as a platform for showcasing the benefits of lossless pooling layer in practice. In particular, a realistic application of the FLPN approach for speeding-up the SR inference in high resolution videos is explored and the applicability of the proposed architecture is once more tested.

Chapter 5

Super-Resolution of Encoded Content

This chapter studies the Super-Resolution (SR) application for Ultra High-Definition (UHD) videos, and in particular, up-scaling from the standard Full High-Definition (HD) resolution to the standard 4K UHD resolution. The main focus in this chapter, unlike the previous chapters is applying SR in encoded content. Moreover, an investigation on application of the presented deep learning restoration model in video compression is presented, which demonstrates promising prospects for encoding high resolution content with low bit-rates and high picture quality.

5.1 Introduction

Emergence of the new standard video formats for television applications has made UHD a trend in broadcasting and streaming services, and more manufacturers are moving towards offering UHD-supported television sets and screens for their customers. Defined first in [5], UHD is a set of standard parameters for video signals used in television production and communication. One of the key aspects of the standard is the spatial resolution of the videos, which can be either 4K UHD (3840×2160) or 8K UHD (7680×4320). The 4K UHD resolution has been the most common UHD format so far, widely used in various broadcasting and streaming platforms.

In today's multimedia ecosystem, there is enough content originally captured and produced in UHD format, to encourage many broadcasting and streaming giants to move

towards adopting this format as their primary platform for providing content. However, one cannot ignore the huge amount of data that has and is still being produced in non-UHD and older video formats, in particular the popular Full HD, which was the dominant standard for several years. Channeling Full HD content through UHD platforms requires various content adaptation and format conversion measures, amongst which spatial up-sampling is one of the crucial ones that converts a video with the native resolution of 1920×1080 to an output video with the 4K UHD resolution, resulting in up-scaling with a factor of 2.

Spatial adaptation is an area where SR can be applied successfully. Given the size of the UHD displays, and the core notion of this video format, which aims at providing high quality picture to the user, it is critical to have efficient and high quality adaptation mechanisms for UHD data creation. Hence SR can be a natural candidate for format conversion. There are, however, two critical issues to consider. One is the complexity of the spatial up-sampling process for UHD content, which was discussed in previous chapter, and will be elaborated further in this chapter, and the other applying SR on encoded content. In previous chapters, the focus was on performing SR on raw uncompressed images (or videos). However, this chapter looks at encoded content, and how SR can affect the quality when deployed on compressed material.

Encoding is another crucial process when dealing with UHD material. UHD videos are extremely large in size, and can take four times more space than the Full HD version when stored in raw format. Therefore finding efficient and effective ways for compressing them remains an ongoing research topic for many, although there has been major breakthroughs in video encoding standardization in recent years by emergence of High Efficiency Video Coding (HEVC). This chapter also looks at encoding UHD content, and proposes an SR-based approach for encoding UHD material using the Full HD version of the same content.

The research goals of this chapter are two-fold. First an efficient high quality SR architecture for up-scaling Full HD videos to UHD resolution is presented and discussed, then a compression paradigm based on the presented SR mechanism is introduced that can be applied for encoding UHD videos using the HEVC-encoded Full HD versions of the same content.

The rest of this chapter is organized as follows. Section 5.2 presents a brief review of the state-of-the-art practices in compressed video SR, as well as the video compression trends for UHD video coding. Section 5.3 presents the proposed SR model for compressed video SR to generate 4K UHD video sequences, with details on model specifications and advantages of deploying such approach. Section 5.4 describes the SR-based compression approach for encoding UHD videos via HEVC-encoded Full HD versions of the content. Section 5.5 reports on the results of various experiments for evaluating the functionality of the presented concepts in this chapter, followed by conclusions in Section 5.6.

5.2 Background

This section provides a summary of the state-of-the-art technology for UHD format from different perspectives. First, practicality of the existing research approaches in applying SR for UHD resolution are discussed, then the issue of up-scaling in compressed domain is raised, which can affect the quality of the SR significantly. Then the mainstream compression schemes for encoding UHD content are outlined, with a focus on HEVC. Finally, the newly emerged paradigms for video compression are briefly discussed, with some of which based on the deep learning concepts.

5.2.1 Video Up-Scaling to UHD

As mentioned previously, spatial resolution adaptation in videos is a critical task in various domains including but not limited to broadcasting and streaming. In particular, scaling to the 4K UHD resolution is an essential and challenging process in many video delivery ecosystems. Application of SR for video up-scaling is an open topic in research community, and numerous contributions have been reported in recent years that aim at deploying deep learning concepts for spatial up-sampling of videos, as reviewed in part in Chapter 2.

The majority of the work in deep learning-based video SR are essentially multi-frame SR approaches, which take several highly correlated frames as input and generate a single target high resolution frame as the output. The output in most of the existing video SR frameworks are single frames. Therefore applying such models for spatial resolution adaptation of the videos requires running the inference process for each target frame

individually. It is needless to say that introducing multi-frame concepts to the SR, which also includes a motion analysis stage, creates a computational overhead to the SR process, hence increasing the complexity and memory requirements for executing the SR process on videos.

In Chapter 4, it was discussed that up-scaling to higher resolutions (HD and beyond) is an extremely power demanding process when relying on Convolutional Neural Networks (CNN). Operating convolutional filters on higher resolutions needs more system memory. Leveraging on that, and the overhead complexity introduced by adopting multi-frame concepts, one can see how impractical it can be to use the existing research-based multi-frame SR approaches for performing video up-scaling to 4K UHD resolution.

The pioneering multi-frame SR approaches such as the ones introduced by Kappeler et al. [61], Makansi et al. [84], Caballero et al. [23], Tao et al. [84], and Sajjadi et al. [94] all promised high quality reconstruction for high resolution video frames, but the basis of the benchmarking for most approaches are research-based low resolution data sets, that can be processes easily on either CPU or GPU platforms. As an example, the method presented in [84] employs the FlowNet 2.0 [53] model for motion analysis of the adjacent frames within a video. FlowNet 2.0 is a very effective optical flow estimation approach using deep learning, that builds the basis of the work described in [84]. However, due to the complexity of the FlowNet 2.0 architecture, it is not plausible to use the best versions of the model when applying it on higher resolutions such as 4K UHD.

Above observations were the source of inspiration for designing efficient SR models for UHD up-scaling in videos, that can provide high picture quality in the reconstructed videos, as well as ability to process videos in a reasonable time using the available technology.

5.2.2 Compressed Video Super-Resolution

Creating high resolution images and videos from low resolution versions of the same content becomes even more challenging, when the content is compressed. The core studies in SR deal with the up-scaling of the raw images and videos, and assume the input to the system are low resolution frames in pixel domain with decent picture quality. However, when considering videos, almost always the content to be considered is already

encoded, and depending on what delivery chain the content has experienced, various artifacts and distortions can be seen in the video signal.

Video encoding is an essential step in video storage and communication systems, that ensures proper compression of the content with an acceptable quality. However visual degradations are inevitable and always exist in encoded content. The level of degradation is of course a function of the encoding scheme and the parameters used in the compression process. Performing SR on encoded video sequences that contain such distortions may lead to further degradations in the high resolution version, causing excessive deterioration of the picture quality. Therefore, designing SR models for compressed videos becomes a more sensitive task that needs to take into account various subtleties concerning video compression processes.

When performing SR on compressed data, a decoding stage needs to be done to parse the bitstream and extract the pixel values corresponding to each frame of the video. The bitstreams associated with compressed videos, that have been encoded with hybrid encoders such as HEVC, contain information about the motion vectors within frames that can be used for motion estimation and compensation step used in different multi-frame SR approaches. Kappeler et al. [60] employed this approach and devised a multi-frame SR model for up-sampling compressed videos, utilizing the motion information coming from the bitstream.

Recent studies in video SR lack a comprehensive analysis of the compression impact on SR quality. Hence it is of major importance to investigate the relation between the picture quality on compressed videos up-scaled to the 4K UHD resolution using modern SR approaches, and the essential compression parameters used in encoding UHD content.

5.2.3 Overview of HEVC

Video encoding is a critical process in the delivery chain, that ensures efficient compression of the data, easing the storage and transmission of videos. There are various standardized encoding schemes that can be used for compressing UHD content. The most recent standard that can effectively compress the content, and is widely used for UHD encoding is HEVC. The HEVC [106] standard (ITU-T H.265 and ISO/IEC 23008-2 [4]) has been developed with the main goal of providing significantly improved video

compression performance compared to its predecessors such as the Advanced Video Coding (AVC) [123] standard (ITU-T Rec. H.264 and ISO/IEC 14496-10 [2]).

The HEVC follows a similar approach as AVC, as video signals are encoded as a sequence of slices, which in many cases one frame constructs exactly one slice. Each slice is then partitioned into non-overlapping Coding Tree Units (CTU), and each CTU can be further partitioned into Coding Units (CU). The content of each CU is efficiently predicted using the previously encoded content. The prediction can be performed either using only the information from the frame currently undergoing encoding (intra-prediction), or using the information from previously encoded frames (inter-prediction).

Three types of slices are used in HEVC, namely "I", in which only intra-prediction can be used, "P", in which a single reference frame can be used for prediction, and "B", in which two frames can be used for prediction. Prediction is performed separately on specific blocks of content by possibly splitting a CU into rectangular or square blocks, referred to as Prediction Units (PU). The process of identifying the optimal prediction for a given PU given the corresponding reference frames is referred to as motion estimation. The output of motion estimation is typically in the form of Motion Vectors (MV), which identify the location in the reference frame where to extract the prediction for the currently predicted block of samples. The collection of the MVs and reference indexes necessary to compute the prediction for a PU is usually referred to as motion information.

The prediction is then subtracted from the original block to obtain the residual samples, which are transformed to an alternative domain before being quantized. Transform and quantization are applied independently to Transform Units (TU), namely square blocks of samples obtained by sub-partitioning the CU following the so-called Residual Quad-Tree (RQT), where a specific range of RQT depths can be used. The quantization stage is where information is lost, meaning that the reconstructed block will be different to the original. HEVC allows the strength of the quantization to be controlled by means of a Quantization Parameter (QP) ranging from 0 to 51, where higher values correspond to a coarser quantization, and resulting in lower quality of the reconstructed sequence for lower bit-rates.

During the HEVC standardization process, three main configurations [19] were defined, which are periodically repeated during the encoding, depending on the access configuration. In the all-intra configuration, all frames are coded as I slices. This is useful for

particular applications such as editing or production, where no inter-frame dependencies are allowed. In the low-delay configuration, the frames are coded in display order. Frames can be encoded as P or B slices, where only frames whose temporal index is lower than the temporal index of the current slice can be used as reference frames. The low-delay configuration is only useful in specific applications where encoding efficiency must be sacrificed to encode with the smallest delays. Finally, the random-access configuration makes use of a more complex coding order where frames are encoded as P or B slices and are compressed in a different order than they are displayed.

5.2.4 Deep Compression and Future Coding Standards

Research in encoding algorithms for image and video compression has also been impacted recently by the advances in application of deep learning. There have been several efforts in finding ways for incorporation of deep learning concepts in compression paradigm, and a new research area called deep compression has emerged. Learning has been applied successfully for predicting and modeling various decision-making algorithms within encoding chains.

Some of the work in applying deep learning for compression include the frame partitioning approach for complexity reduction of intra-prediction modes in HEVC proposed by Li et al. [75], a deep learning-based approach for selecting efficient intra-prediction modes in HEVC proposed by Laude and Ostermann [70], an arithmetic coding method based on neural networks introduced by Song et al. [105], rate-distortion modeling for HEVC using CNNs by Xu et al. [124], and estimation of rate control parameters for HEVC using deep learning models by Santamaria et al. [96].

In addition to application of deep learning for enhancing existing compression algorithms such as HEVC, there are efforts in devising novel paradigms as future coding standards. With regard to encoding UHD content, the Versatile Video Coding [6] also addresses the issue of further efficiency in compression and higher picture quality.

Amongst the promising novel approaches in video coding, SR can also be considered as a candidate. There have been several attempts for adopting SR concepts as a key dynamic for creating image and video compression tools. Afonso et al. [8, 9] proposed low complexity video compression approach using spatio-temporal resolution adaptation,

which dynamically re-samples the video during encoding and encodes either the original or the re-sampled frames based on a quantization-resolution decision. Georgis et al. [39] used a rather similar approach exploiting joint down-sampling/up-sampling mechanisms to provide a low bit-rate video compression framework. Liu and Cui [82] also contributed in applying SR in compression by down-sampling the videos prior to encoding and re-sampling them back to the original spatial resolution at the decoder side.

The above-mentioned research were the inspiration for proposing an HEVC-based video compression work-flow for encoding UHD content using high quality deep learning-based SR models for complexity reduction and low bit-rate encoding of the video sequences in 4K UHD resolution.

5.3 High Quality Up-Scaling from HD to UHD

This section describes the detailed deep learning-based SR model designed for spatial up-sampling of the HEVC-compressed videos from the Full HD resolution of 1920×1080 to 4K UHD resolution of 3840×2160. The core CNN architecture is based on the successful EDSR model for still image SR proposed by Lim et al. [77]. The model is adapted in terms of functionality to achieve a more efficient performance in terms of computational complexity, so it can be applicable for UHD resolution.

An image-based approach is selected for video SR, in which every frame of the video is up-sampled independently from the adjacent frames. Application of multi-frame methods were avoided for the reasons explained in 5.2. In principle, multi-frame approaches tend to create extra complexity overhead to the SR framework, and eventually perform the up-scaling frame-wise. Hence multi-frame approaches, although more effective in terms of picture quality, can be impractical to use. However employing a very high quality still image SR for videos can also provide a favorable picture quality, but more importantly the process is not as time consuming as the multi-frame approaches, and the up-sampling for UHD content can be handled on existing hardware technology.

In this regard, the EDSR can be a suitable candidate for this purpose, as it can promise high quality picture quality and image reconstruction. Moreover, proper modifications of the architecture can result in significant complexity reduction in network inference with minor impact on the quality performance, thus leading to a robust SR approach for

compressed videos. In addition, extra measures are taken in the training of the proposed architecture in order to be able to handle a wide range of compression artifacts that exist in an encoded video. Therefore, the model is adapted to cope with compressed content effectively. In the following, the ESDR architecture is presented, and the proposed modifications based on the lossless pooling approach presented in Chapter 4 are described for improving the efficiency of the model, along with discussions on specific training data preparation for coping with compressed content.

5.3.1 Baseline Deep Super-Resolution Network

EDSR network was proposed by Lim et al. [77] in the context of an image SR challenge in 2017, and the model succeeded to perform as the best image restoration model in terms of picture fidelity and the perceptual quality of the content. The model is inspired by the generative network of the GAN-based approach proposed by Ledig et al. [72], taking advantage of the innovative learning architecture defined by the ResNet model [46].

Since this chapter aims at up-scaling from Full HD resolution to 4K UHD, the focus is on the EDSR architecture designed for the scaling factor of 2. The baseline architecture is depicted in Figure 5.1, with details on the number of filters in each layer as well as the size of the filter kernels for each convolutional layer. Accordingly the input image is fed to network as it is with no pre-processing (bi-cubic interpolation), and all the processes take place in the original spatial resolution. The actual scaling of the image happens in the last layer using the periodic shuffling operation defined in Chapter 2.

The network comprises 32 residual blocks, and each residual block includes a convolutional layer, followed by ReLU activation, followed by a second convolutional layer, and followed by a multiplier. The resulting output is added to the input of the block to create the residual structure. The multiplier acts as a scaling stabilizer to ensure the stability of the training procedure given the excessive number of parameters in the model. The baseline network presented as EDSR does not contain any batch normalization layers to avoid removal of range flexibility, and significantly reduce the memory requirements of the residual network.

The training of the original $\times 2$ EDSR model is done by Adam [65] optimizer while gradually decreasing the learning rate in the training process. The DIV2K [10] training

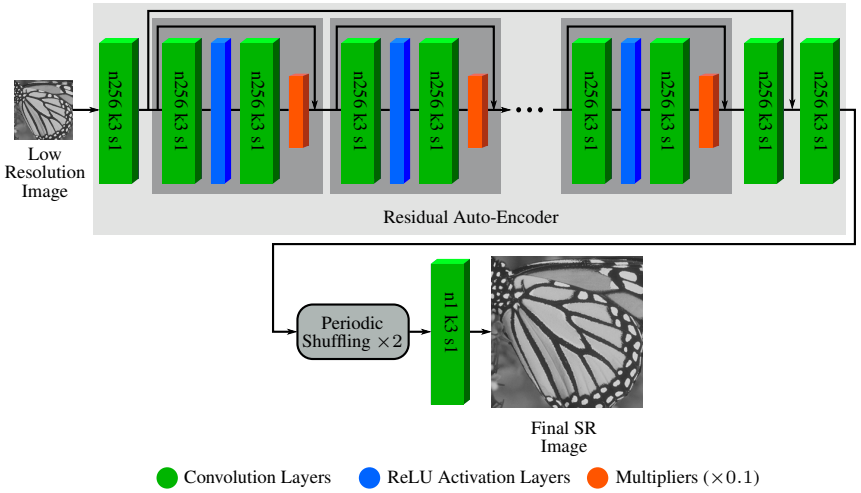


FIGURE 5.1: Baseline architecture for EDSR with scaling factor of 2. The parameters n, k, and s specify the number of filters, kernel size, and stride value for every layer.

images are used as the training data, and data augmentation is employed by randomly selecting cropped low resolution patches of size 48×48, and performing random flips and rotations to the images. With regard to the cost function, the authors of the EDSR model adopted ℓ_1-norm instead of ℓ_2-norm, as they witnessed a slight improvement in the performance of the trained model in terms of picture quality when choosing ℓ_1-norm.

As per the original model designed by Lim et al. [77], the model takes a three-channel (colored) image as input, and generates a three-channel high resolution image as the output. In this chapter, however, the model is considered to be operating for single-channel (grayscale) images as depicted in Figure 5.1 to reduce the complexity of the up-scaling and compression work-flow in the next sections. Moreover, given that most of the visual and perceptual information lies within the luminance channel, this approach is not expected to have a major impact on the quality of the SR.

5.3.2 Modified ESDR for Compressed UHD Up-Scaling

In designing spatial adaptation models for compressed content, two aspects need to be taken into account: The complexity of the processes for performing scaling operation, and the quality considerations imposed by the compression artifacts in the content. It was mentioned that the EDSR model is chosen as the core baseline for this endeavor. However, EDSR cannot be used as it is for scaling HD content to the desired UHD resolution. The huge number of parameters in the EDSR, and the excessive number of convolutional filters in the network architecture demand significant amount of memory.

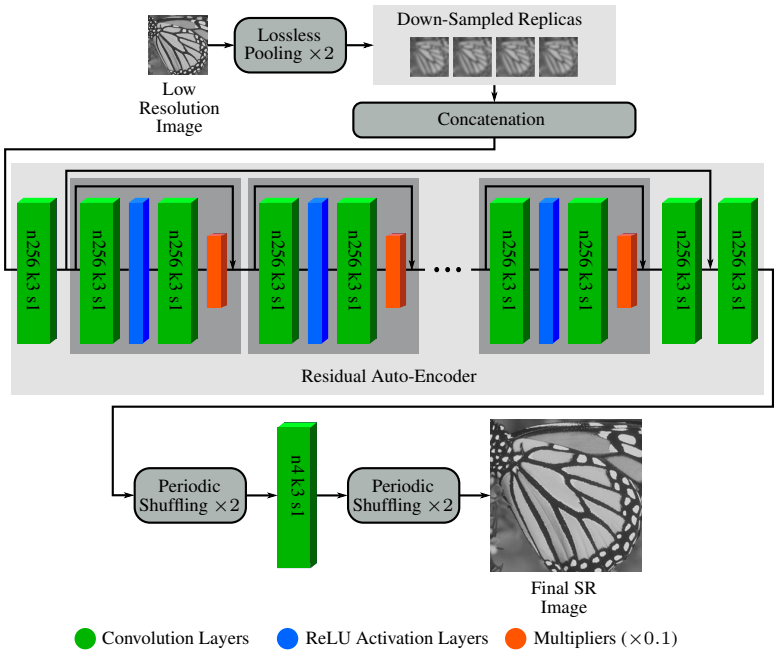


FIGURE 5.2: Modified EDSR with lossless pooling layer for low complexity Full HD to 4K UHD scaling. The parameters n, k, and s specify the number of filters, kernel size, and stride value for every layer.

Even though the majority of the operations in this structure take place in the native low resolution space, the native input resolution is 1920×1080 for HD to UHD adaptation, which is still a very high resolution, leading to impractical memory requirements and complexity overhead.

Complexity Considerations for SR Model

In order to address the model complexity issue, the lossless pooling concept introduced in Chapter 4 is employed to reduce the memory requirements and computational load of the model, and create a realistic UHD up-scaling framework. In particular, the FLPN structure described in Section 4.3.1 is adopted, that ensures complexity reduction of SR models by decreasing the input spatial size at an early stage of the SR process. The lossless pooling layer with a factor of 2 is integrated in the EDSR baseline architecture as depicted in Figure 5.2.

The baseline EDSR depicted in Figure 5.1 takes a single-channel image as its input signal. In the proposed modified EDSR depicted in Figure 5.2, the input signal is first fed to the lossless pooling layer, which creates a four-channel image, with each channel containing a replica of the original input image with half the spatial resolution in each dimensions. Accordingly, an input grayscale Full HD frame with resolution

1920×1080×1 will generate a tensor with the shape 960×540×4 after going through the lossless pooling. The mentioned tensor is then processed by the baseline EDSR model. The only change in the EDSR model is the last layer, which now has 4 convolutional layers instead of the original 1 presented in Figure 5.1. The output to this final layer is a tensor with the shape of 1920×1080×4, which is then fed to one last periodic shuffling block, resulting in the desired high quality high resolution output frame with the resolution 3840×2160×1.

In theory, the proposed architecture requires quarter of the memory that the original EDSR requires. Given that the lossless pooling layer does a scaling with a factor of 2, the number of pixels throughout the network for all the layers will be reduced to quarter of the case with the original EDSR, leading to significant savings in memory and speed-ups in computing the SR image. Chapter 4 reported on the picture quality when using lossless pooling layer, however there are more experiments in this chapter focusing on the performance of the EDSR, that will be presented in the next sections.

Quality Considerations for SR Model

Performing SR on compressed data is typically done by first decoding the bitstream, then performing the advanced scaling process on the pixels parsed from the compressed data. The process is essentially the same as SR on raw images and video in terms of the operation space. However, the key difference is the quality of the input, that has undergone compression; an encoding procedure with parameters that may be unknown to the SR model. In image and video SR for raw data, it is often assumed that the input low resolution content is of high quality. Even if the low resolution content is the result of a down-scaling process, the existing artifacts would be ringing and aliasing that can be addressed during the training by replicating such scenarios in the training data.

The artifacts introduced by compression, on the other hand, can be more elaborate and contain different categories of distortions such as contouring, blockiness, or posterizing. Hence the usual training process applied for still image and video SR may not fully cover the different distortion possibilities caused by compression. Therefore, it is critical to create a more inclusive training data set for SR models that aim at handling compressed data. To achieve this goal, an extra pre-processing step is required for creating the training data set.

The proposed data set used for training the modified EDSR for compressed SR is DIV2K [10] similar to the original EDSR. However, instead of simply down-scaling the high resolution content for obtaining the network input training data, an extra compression stage is performed on images to introduce encoding artifacts to the training set. The DIV2K data set images come as PNG format [3], which is essentially a lossless compression platform retaining the visual fidelity of the content. DIV2K data set also comes with already down-scaled versions of the images for SR training purposes. There are two down-scaled versions: One created by bi-cubic interpolation, and one created by unknown down-scaling and down-grading operations that may contain more distortions. The second version, however, deals mainly with different levels of blurriness and is not a very suitable option for compression scenarios.

To achieve the proper data set for compressed content SR, the high resolution DIV2K is down-sampled using bi-cubic interpolation, then compressed using a codec with a random compression ratio, so variety of compression cases with numerous distortion scenarios will be available as the training set. Two encoders were used for compressing the down-scaled images: HEVC in all-intra mode, which is applicable for still image coding, and JPEG [51].

The aim is to design an SR method for videos encoded by HEVC. In particular, the random-access profile of HEVC is of main interest, given the popularity and the applicability of this configuration in terms of coding efficiency. As per design goals of the model, it makes sense to use HEVC codec to compress the training data, so a more coherent training is performed, with the training data matching the envisaged application scenarios. However, all-intra profile in HEVC can provide a much better picture quality when compared to the random-access profile, and disparity in quality comes from the fact that in random access profile, majority of the frames take advantage of the inter-prediction, which can significantly improve the compression efficiency for the price of degraded quality. Hence it seems to be unfair to have all-intra mode for the training, and random-access in the testing phase.

In order to resolve this, JPEG was chosen to be used together with HEVC all-intra to balance the quality of the training set for compressed SR model based on the modified EDSR. It is known that HEVC all-intra is a superior encoder than the JPEG in terms of picture quality and compression efficiency. However, given that random-access profile

cannot be applied to the training set, which comprise of still images, JPEG can used to mimic the HEVC random-access profile and provide a lower quality compression compared to HEVC all-intra. In principle, HEVC all-intra with several QPs, along with JPEG with a range of Compression Ratios (CR) construct a set of encoding scenarios that can provide a comprehensive range of possible distortions that can occur during picture encoding. Therefore, using this set on the training data can simulate the essential condition for dealing with compressed data.

For the case of HEVC all-intra, the following QPs were selected: 18, 22, 26, 30, and 34. The five QPs create five levels of compression quality. The chosen QP values correspond with the range of compression used in the test data. Moreover, the selected values lie within the range of the typical QP values for various realistic scenarios such as broadcasting or streaming services. For the case of JPEG, the CR values can be between 0 to 100, with 0 coding the most aggressive type of compression, and 100 coding a near lossless case. In order to have a reasonable range that works well next to HEVC, the following CR values were chosen based on extensive evaluations: 50, 60, 70, 80, and 90.

The above-mentioned compression scenarios create a list of 10 options for encoding the input training data, 5 using the HEVC all-intra configuration and 5 using JPEG. Therefore, in order to create the final data set, the high resolution DIV2K training set is first down-sampled by a factor of 2 using bi-cubic interpolation, then 700 out of the 800 images in the data set are randomly grouped into 10 classes. Each class of down-scaled images are encoded using one of the existing coding scenarios, and the remaining 100 images are not encoded, to be able to handle uncompressed data as well, and increase the range of the training. The ground truth data, however, remain as they are with the highest quality, so that the network can map the compressed content to the best available quality. Figures 5.3 and 5.4 show examples of the ten devised compression scenarios and their impact on the picture quality for the sample *Butterfly* image. The original *Butterfly* image has a spatial resolution of 256×256. The image is down-scaled to 128×128 (shown as "Uncompressed" in Figures 5.3 and 5.4), and then compressed using HEVC or JPEG. The PSNR values, along with the visual quality of the different cases demonstrate the wide range of compression quality that can be achieved by these ten scenarios.

FIGURE 5.3: An example illustrating the impact of HEVC all-intra compression on *Butterfly* image for different QPs.

FIGURE 5.4: An example illustrating the impact of JPEG compression on *Butterfly* image for different CRs.

For data augmentation purposes, all the 800 images are rotated 90, 180, and 270 degrees, and flipped horizontally and vertically, resulting in 6400 images in the training set. Images are then partitioned into non-overlapping patches of 48×48 low resolution training samples (96×96 high resolution samples). Consequently a set of 1.8 million

training samples are generated that are suitable for compressed SR training for the scaling factor of 2. Using a batch size of 128 and the cost function introduced in Chapter 2, Equation 3.7, the proposed modified EDSR network depicted in Figure 5.2 is trained. Adam [65] optimizer is employed with the starting learning rate of 0.0001, β_1 of 0.9, β_2 of 0.999, and ϵ of 10^{-8}. The learning rate is reduced by a factor of 2 every 2 epochs ($\sim$200,000 iterations).

In terms of network inference for video SR, every Full HD sequence is first decoded, then the frames are fed to the network independently to create the set of 4K UHD frames associated with the original video. With regard to the different color channels scaling, only the luma channel is up-scaled using the trained CNN, and the chroma signals are scaled using bi-cubic interpolation. This decision is made to reduce the complexity of the process with little impact on the quality, as the crucial perceptual information lies within the luminance pixels [30, 62, 101, 129].

5.4 Compression using Super-Resolution

Previous section described the adopted approach for generating high quality 4K UHD videos from Full HD content. If the quality of the SR is satisfactory, the approach can be considered as a compression tool for encoding UHD content. In Section 5.2.4, some of the innovative approaches in video coding were covered, and application of SR in compression was amongst them. In this regard, this section introduces a video compression pipeline for encoding UHD videos using the deep learning-based SR model for compressed videos described in previous section.

Spatial down-sampling of video signals can be considered as a compression measure, as it leads to reduction of number of pixels, and consequently the content volume and the bit-rate. Therefore, having a high quality SR mechanism that can reconstruct the original video from the down-sampled video with high fidelity can lead to creation of a codec structure. If this concept is coupled with application of HEVC for encoding the down-sampled content, then the work-flow would be complete.

Figure 5.5 depicts the architecture of the codec for UHD compression. The input to the encoder side is of course a 4K UHD video sequence in raw format. Any of the YUV pixel sampling formats can be applied, but the 4:2:0 sampling is considered throughout this

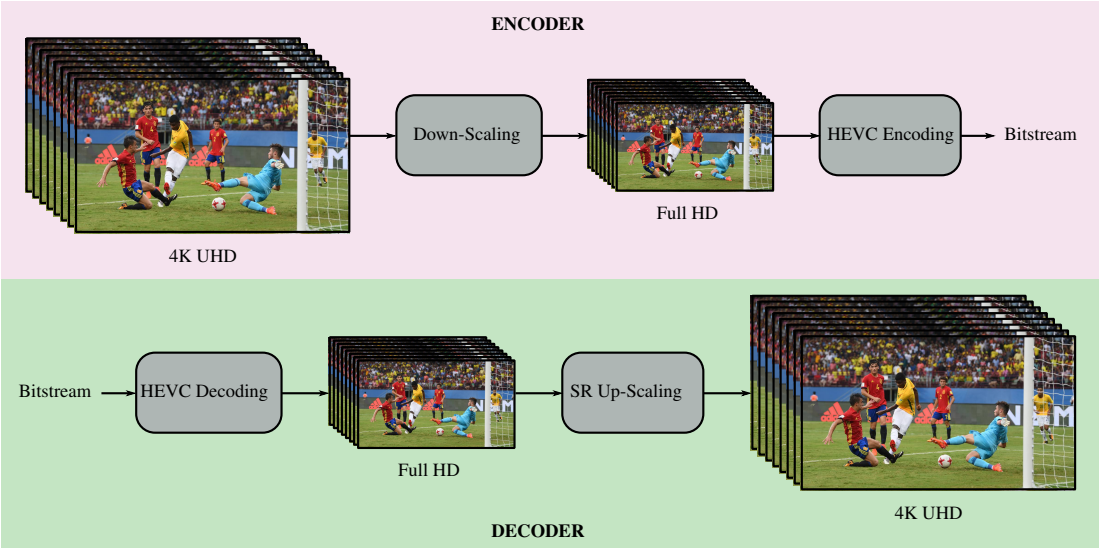


FIGURE 5.5: Structure of the encoder and decoder work-flows for the proposed SR-based UHD codec.

chapter. The sequence goes through a spatial down-sampling process and is converted to a sequence with Full HD resolution. Bi-cubic down-sampling is chosen, as it can provide a reasonable performance in terms of both picture quality and computational complexity. The resulting sequence is then encoded using HEVC encoder, and the bitstream is obtained. This concludes the encoder work-flow, which is a straightforward process. In terms of HEVC encoding, the common test configurations [19] for random-access profile is adopted, and the QP value acts as the main encoding parameter.

With respect to the decoder, an HEVC decoder is in order to parse the bitstream, followed by the high quality SR mechanism for up-scaling the encoded video back to the original 4K UHD resolution. Given that the encoding UHD content is significantly more complex than encoding Full HD content in terms of computation power, the proposed approach saves time in compression process. The price to pay is the decoder complexity which is impacted by the heavy SR process. The complexity of SR, however, can be dealt with by application of powerful GPU platforms. Having said that, perhaps the main theoretical aspect in studying this approach is synthesizing a function that can map an encoded UHD video with a certain QP, to another sequence with the same picture quality obtanied by encoding Full HD version of the same video up-scaled using the proposed SR model.

Compressing any video sequence with the UHD resolution using the HEVC random-access profile with a certain QP results in a particular picture quality that can be

represented by a PSNR value. Using the SR-based approach for video compression, also requires a QP value for encoding the down-sampled Full HD sequence. The resulting bitstream will be decoded and up-scaled to UHD with a certain quality represented by PSNR. One of the research goals in this section is to find a correspondence between the QP used in UHD encoding and the QP used in Full HD encoding. In other words, there can be a function as the following that can represent this QP mapping:

$$Q^{UHD} = m_q(Q^{HD}) \tag{5.1}$$

where Q^{UHD} represents the QP value for encoding the UHD sequence directly with HEVC in random-access mode, Q^{HD} represents the QP value for encoding the UHD sequence by first down-scaling it, then encoding it using HEVC random-access, and m_q represents the mapping function that relates the two QP values and ensures the same picture quality in terms of PSNR for both compression approaches. Figure 5.6 depicts the graphical interpretation of the QP mapping concept introduced in this section for achieving the same picture quality represented by the PSNR metric.

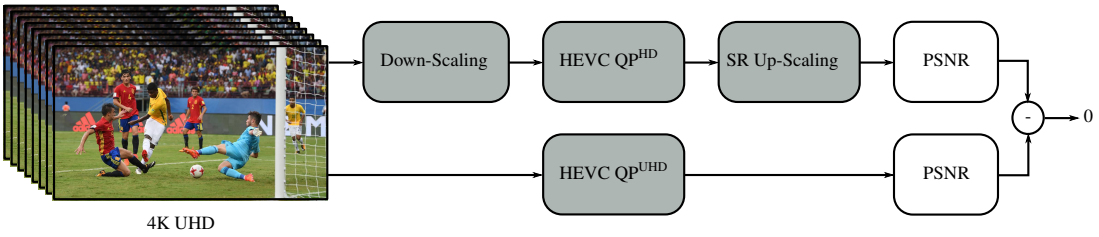


FIGURE 5.6: QP mapping for the conventional UHD HEVC encoding and SR-based HEVC encoding to achieve the same picture quality (PSNR).

5.5 Experiments

The training procedure for the compressed video SR network was described thoroughly in Section 5.3.2. The trained network aims at performing high quality $\times 2$ up-scaling in compressed videos with Full HD (1920×1080) resolution to achieve 4K UHD (3840×2160) resolution sequences. The model needs to be tested from different perspectives in order to ensure the effectiveness and functionality of the approach in the desired domain.

The first set of experiments focus on performance of the proposed modified EDSR model as a stand-alone SR engine. The picture quality in the reconstructed output signal needs to be evaluated and compared with the state-of-the-art approaches. More importantly,

the complexity of the approach must be examined by looking at the computation time for generating UHD content from Full HD frames. This set of experiments can be considered as a benchmarking activity for the proposed modified EDSR, regardless of the efforts put into adaptation of the model for compressed videos, and only taking into account the CNN architecture and the proposed deep learning structure.

The second set of experiments concentrates on the performance of the approach on the compressed content, and examining the functionality of the QP mapping concept proposed in Section 5.4. In addition, considerations on performance of the proposed SR-based compression scheme in terms of bit-rate in comparison with direct HEVC encoding of the UHD content are taken into account.

5.5.1 Evaluation of Modified EDSR

The modified EDSR model presented in this chapter, is based on the single image SR model presented in [77]. Given that this section mainly focuses on the evaluation of the proposed model with considerations on the architecture and functionality of the network structure, single image data sets are used for evaluation. The compression is ignored here, although the network is trained using the compressed material as described in previous sections. The DIV2K validation set and Ultra Eye Tracking (UET) data set [89], also used in Chapter 4, are adopted as the main test sets, as the focus of this chapter is on high resolution content. The state-of-the-art methods including SRCNN [30], ESPCN [101], FSRCNN [32], REDNet [85], as well as original EDSR [77] are tested against the proposed architecture.

The focus of the experiments are on scaling factor of 2, and both picture quality and computation time are taken into account in the evaluations. Specifically, the UET data set is of interest, as it can resemble the Full HD to UHD scaling, which is the main goal in this chapter. The concept of lossless pooling was evaluated comprehensively in Chapter 4, however since the baseline architecture is different in this chapter (EDSR rather than REDNet), it is important to compare the picture quality in the proposed architecture, with well-known methods. It is predictable that the proposed model may underperform slightly when compared with the original EDSR, however the complexity of the approach is expected to be significantly lower compared to EDSR, and that is to be tested.

TABLE 5.1: Quality and complexity analysis of the proposed EDSR-based method and state-of-the-art approaches for the scaling factor of 2.

		Bicubic	**SRCNN**	**ESPCN**	**FSRCNN**	**REDNet**	**Mod-EDSR**
DIV2K	PSNR	32.36	34.18	34.42	34.63	35.28	35.72
	SSIM	0.9020	0.9267	0.9293	0.9322	0.9371	0.9406
	GPU time	-	0.7	3.3	1.2	1.8	62.8
UET	PSNR	36.10	37.92	38.12	38.24	38.80	39.21
	SSIM	0.9369	0.9534	0.9551	0.9570	0.9593	0.9614
	GPU time	-	1.2	8.4	2.5	4.1	167.9

Table 5.1 presents the results of the tests carried out on the state-of-the-art approaches other than original EDSR in comparison with the proposed modified EDSR. All the tests were run on a GeForce GTX 1070 GPU with 8GB internal memory. It is evident that for scaling factor of 2, and on both data sets, the proposed approach outperforms the other methods as expected, and there are significant improvements in the picture quality in terms of both PSNR and SSIM compared to the next best model, which is REDNet. It was also expected that the model has a higher complexity compared to the rest, given the excessive number of parameters and filters in the EDSR architecture.

With regard to the complexity, though, it is critical to compare the proposed structure with the original EDSR, as the proposed model is essentially a modified improved version of the EDSR, aiming at achieving faster performance for UHD scaling. With respect to the EDSR quality performance, it is expected that the proposed modified EDSR, falls behind original EDSR in terms of PSNR and SSIM, due to the introduction of the lossless pooling structure to the network architecture. According to the reported results in Chapter 4, adding lossless pooling layer in the FLPN format (cf. Section 4.3.1) to a baseline SR network, may result in a slight reduction in the PSNR and SSIM values, while reducing the computation time of the model significantly for higher resolution reconstructions. Consequently, similar behavior is expected to happen for EDSR.

Table 5.2 presents the results of the tests performed for complexity monitoring of the proposed modified EDSR (Mod-EDSR) on different GPU architectures in comparison with the original EDSR (Org-EDSR). Two platforms were used for running the tests, as applying SR on UHD content can become very memory consuming, and the processing time may vary significantly in different platforms depending on the capability of the GPUs. In fact, some architectures may not even be able to run successfully on some GPUs. As an example, the original EDSR cannot operate on any of the available hardware for UHD scaling, as the GPUs run out of memory during the network inference

TABLE 5.2: Quality and complexity analysis of the proposed EDSR-based method and original EDSR for the scaling factor of 2, on different GPU platforms.

		Org-EDSR	**Mod-EDSR**
DIV2K	PSNR	35.78	35.72
	SSIM	0.9411	0.9406
	GeForce GTX1070	200.1	62.9
	Tesla K80	49.9	16.3
UET	PSNR	-	39.21
	SSIM	-	0.9614
	GeForce GTX1070	-	167.9
	Tesla K80	-	37.3

for performing up-scaling from Full HD resolution to 4K UHD. Therefore, only the validation set of DIV2K data set was used for comparing the performance of the modified EDSR and original EDSR.

The tests were performed on the following hardware: GeForce GTX 1070 with 8GB internal memory, and Tesla K80 with 12GB internal memory. The first setting is the weakest, in terms of memory resources, and as reported in Table 5.2, the modified EDSR has slower inference in terms of computation time on this platform. Original EDSR, however, fails to operate on both platforms on UHD data set. On the DIV2K data set (HD resolution), the modified EDSR significantly improves the memory requirements of the model and reduces the computation time when compared to its EDSR baseline model. In terms of picture quality, the original EDSR outperforms the proposed modified model slightly, which was expected, however the great complexity and memory requirements of the original model make it absolutely impractical to be deployed for UHD scaling.

5.5.2 Evaluation of QP Mapping

The main scope of this chapter has been generating SR models for compressed videos. In this regard, this section evaluates the performance of the proposed deep learning-based up-scaling model on actual encoded videos. The same modified EDSR model tested previously on still images is used as the SR engine, and the model is triggered on every frame of the video sequences, to perform high quality up-sampling. 12 video sequences are used as the test data, all of which are the well-known test content in video compression research domain. Table 5.3 summarizes the characteristics of the videos used in the evaluations.

TABLE 5.3: Detailed characteristics of the video sequences used for evaluation of QP mapping.

Source	Name	Resolution	Rate	Frames	Pixel Sampling
BBC [122]	Book	3840×2160	50	500	YUV 4:2:0
	CalendarAndPlants	3840×2160	50	500	YUV 4:2:0
	MenAndPlants	3840×2160	50	500	YUV 4:2:0
	ParkAndBuildings	3840×2160	50	500	YUV 4:2:0
	Vehicles	3840×2160	50	500	YUV 4:2:0
Digiturk [1]	Beauty	3840×2160	120	600	YUV 4:2:0
	Bosphorus	3840×2160	120	600	YUV 4:2:0
	HoneyBee	3840×2160	120	600	YUV 4:2:0
	Jockey	3840×2160	120	600	YUV 4:2:0
	ReadySteadyGo	3840×2160	120	600	YUV 4:2:0
	ShakeNDry	3840×2160	120	600	YUV 4:2:0
	YachtRide	3840×2160	120	600	YUV 4:2:0

In order to assess the functionality of the QP mapping concept presented in Section 5.4, every video sequence undergoes three separate processes:

1. Each sequence is encoded directly using HEVC random-access profile under common test configurations with the following QPs: 22, 24, 26, 28, 30, 32, 34.

2. Each sequence is down-scaled to the Full HD resolution of 1920×1080 using bi-cubic interpolation, then encoded using HEVC random-access profile under common test configurations with the following QPs: 16, 18, 20, 22, 24, 26, 28. The resulting encoded sequences are then up-scaled to the original UHD resolution using the modified EDSR approach.

3. Similar to above, except the up-scaling is done by bi-cubic interpolation to create a baseline for experiments.

All the HEVC encodings were done using the HM Reference Software implementation [116]. Above mechanisms result in multiple versions of the same content, some of which obtained by direct encoding of the UHD video, others obtained by re-scaling and encoding process proposed in this chapter. It is worth noting that the SR process is performed only on the luminance signal (Y channel), and the chrominance signals are up-scaled using bi-cubic interpolation for simplicity. Therefore the reported PSNR values in the following correspond to the PSNR on the Y channels only, as the focus of the activities have been on the high quality luma scaling, although the concept can be easily extended to chroma channels, too.

All the various versions of the content obtained by the three described processes result in specific picture qualities, when compared to the original uncompressed raw footage.

TABLE 5.4: PSNR values for UHD test sequences created by direct HEVC encoding.

	UHD QP	**22**	**24**	**26**	**28**	**30**	**32**	**34**
UHD Encoding	Book	45.96	45.37	44.70	44.00	43.20	42.30	41.38
	CalendarAndPlants	45.29	44.53	43.75	42.98	42.18	41.31	40.43
	MenAndPlants	44.89	44.12	43.31	42.48	41.62	40.71	39.79
	ParkAndBuildings	41.66	40.73	39.90	39.11	38.27	37.34	36.36
	Vehicles	40.34	39.00	37.89	37.03	36.30	35.48	34.58
	Beauty	36.83	35.75	35.17	34.86	34.66	34.52	34.39
	Bosphorus	43.17	42.64	42.08	41.50	40.88	40.21	39.51
	HoneyBee	39.45	39.13	38.98	38.85	38.68	38.44	38.18
	Jockey	40.33	40.03	39.82	39.61	39.36	39.02	38.63
	ReadySteadyGo	41.52	41.09	40.61	40.08	39.45	38.71	37.93
	ShakeNDry	39.48	38.94	38.40	37.74	37.02	36.22	35.43
	YachtRide	42.60	41.79	40.90	39.96	38.98	37.97	37.02
	Average	41.79	41.09	40.46	39.85	39.22	38.52	37.80

TABLE 5.5: PSNR values for UHD test sequences created by compressed SR from encoded Full HD using modified EDSR.

	HD QP	**16**	**18**	**20**	**22**	**24**	**26**	**28**
Compressed Scaling (Modified EDSR)	Book	45.97	45.59	45.14	44.60	44.08	43.26	42.48
	CalendarAndPlants	45.08	44.61	43.98	43.38	42.63	41.90	41.06
	MenAndPlants	45.00	44.49	43.93	43.27	42.55	41.76	40.95
	ParkAndBuildings	38.58	38.19	37.74	37.19	36.52	35.76	34.96
	Vehicles	36.07	35.71	35.33	34.84	34.25	33.61	32.92
	Beauty	36.93	36.59	36.36	36.06	35.93	35.70	35.58
	Bosphorus	43.49	43.13	42.72	42.13	41.56	40.82	40.16
	HoneyBee	39.65	39.50	39.29	39.19	39.05	38.74	38.49
	Jockey	40.72	40.56	40.40	40.12	39.79	39.50	39.14
	ReadySteadyGo	41.63	41.34	40.89	40.44	39.80	39.16	38.25
	ShakeNDry	39.93	39.63	39.09	38.59	37.93	37.05	36.20
	YachtRide	41.76	41.38	40.82	40.19	39.33	38.52	37.48
	Average	41.23	40.89	40.47	40.00	39.45	38.82	38.14

TABLE 5.6: PSNR values for UHD test sequences created by compressed scaling from encoded Full HD using bi-cubic interpolation.

	HD QP	**16**	**18**	**20**	**22**	**24**	**26**	**28**
Compressed Scaling (Bi-cubic Interpolation)	Book	44.72	44.37	43.94	43.42	42.18	42.10	41.32
	CalendarAndPlants	42.68	42.34	41.95	41.49	40.97	40.37	39.72
	MenAndPlants	43.12	42.17	42.24	41.67	41.02	40.30	39.53
	ParkAndBuildings	35.45	35.28	35.07	34.80	34.46	34.05	33.58
	Vehicles	33.15	32.98	32.79	32.55	32.24	31.90	31.51
	Beauty	35.59	35.27	35.04	34.85	34.72	34.59	34.47
	Bosphorus	43.00	42.60	42.16	41.64	41.02	40.33	39.64
	HoneyBee	39.22	39.05	38.93	38.81	38.65	38.44	38.19
	Jockey	40.15	39.97	39.81	39.63	39.39	39.10	18.73
	ReadySteadyGo	40.45	40.21	39.92	39.54	39.05	38.45	37.77
	ShakeNDry	39.36	39.04	38.69	38.17	37.49	36.69	35.81
	YachtRide	40.36	40.05	39.67	39.13	38.45	37.71	36.91
	Average	39.77	39.49	39.18	38.81	38.36	37.84	37.26

Tables 5.4-5.6 summarize the PSNR values for all the generated sequences. Table 5.4 reports the PSNR values for the case of direct UHD encoding using HEVC. This is considered to be a high quality baseline for the compressed UHD videos, given that the content is encoded with HEVC with realistic QPs. Table 5.5 outlines the picture quality when performing scaling and encoding, or in other words, performing compressed SR.

The down-scaled Full HD content is encoded using HEVC, and the decoded sequences are scaled up using modified EDSR model. The average PSNR values show that the results are in the same range of what was achieved by direct UHD encoding, and there is a correspondence between the picture quality in these two scenarios. Consequently, the QP mapping concept is functioning successfully. Table 5.6 reports the picture quality when performing scaling and compression, except the up-scaling is performed by bi-cubic interpolation, hence a lower bound reference is created using this approach. The average PSNR values in this case are far smaller than the case presented in Table 5.5, which is another proof on effectiveness of the modified EDSR model for performing high quality up-scaling for compressed SR.

In order to have a better understanding of the picture quality when performing compressed SR, Figure 5.7 can be referred to, that demonstrates the average PSNR values for the 12 test sequences undergone the three described scenarios. The PSNR curves are drawn against the QP values for each scenario, and it is vivid that as expected there is a significant improvement in the picture quality when using modified EDSR instead of bi-cubic interpolation for compressed SR. More importantly the results correspond with

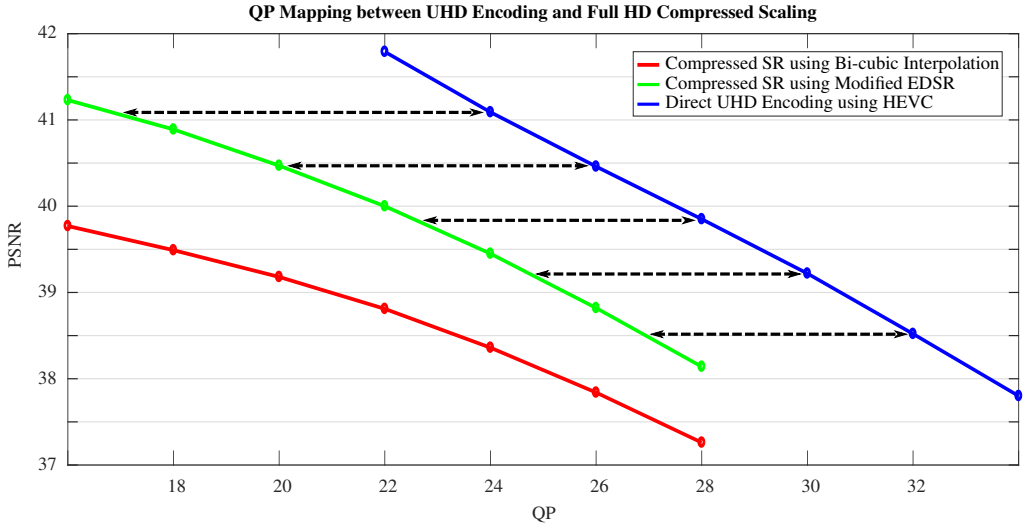


FIGURE 5.7: The correspondence between the PSNR values obtained from the three tested scenarios including the direct HEVC encoding of UHD content, and compressed scaling from encoded Full HD versions using modified EDSR and bi-cubic interpolation.

TABLE 5.7: Interpolated QP mapping between the QP values used in direct HEVC encoding of the UHD content in random-access profile (Q^{UHD}), and the QP values used in HEVC encoding of the down-scaled Full HD content to be up-scaled using modified EDSR (Q^{HD}).

$\mathbf{Q^{UHD}}$	24	26	28	30	32
$\mathbf{Q^{HD}}$	17	20	23	25	27

the case of direct UHD encoding depicted with the blue curve. It is, however, evident that in order to achieve this correspondence different QP ranges need to be used for the direct UHD encoding and the Full HD encoding in case of compressed SR. The horizontal lines in Figure 5.7 demonstrate the QP mappings pursued in this chapter. Similar PSNR values can be achieved by direct UHD encoding and compressed SR, if certain QP values are used for the HEVC encoding of the UHD and Full HD video sequences.

Although the QP values are limited in this set of experiments, it can be concluded that a set of QP mappings can be obtained from the results, summarized in Table 5.7. For high QP values the mapping between the Full HD and UHD values is expected to be a simple offset, whereas for smaller QP values a more non-linear behavior is expected. Three points corresponding to $Q^{UHD} = \{24, 28, 32\}$ are selected to fit a second order polynomial curve with least squares quadratic regression, to represent the possible QP mapping of HEVC encoding and compressed SR using modified EDSR. From the mappings interpolated from the Figure 5.7, the following curve-fitting can be formulated, which is the solution to the QP mapping equation 5.1.

$$Q^{UHD} = 25.7 - 0.67Q^{HD} + 0.03(Q^{HD})^2 \tag{5.2}$$

When investigating the spatial scaling of compressed content, and devising encoding schemes and QP mapping, it is only fair to also consider the impact of each scenario on the bit-rate when evaluating the performance. Each compressed video sequence is encoded with a certain bit-rate, which represents the average number of bits for every second of the sequence. Similar to the PSNR values that change based on the choice of the QP, the bit-rate is also dependent on the QP value. Moreover, the bit-rate has a direct relation with PSNR, in which high bit-rates correspond to high PSNR values.

Figure 5.7 demonstrated that the compressed SR using modified EDSR can provide the same performance in terms of picture quality compared to the direct HEVC encoding. It is also crucial to monitor the same impact on the bit-rates. Figure 5.8 depicts the rate-distortion curves for the three discussed test scenarios. The results show that the compressed SR can provide the same range of bit-rate as the direct HEVC encoding. In fact, there is a slight gain in applying compressed SR, which is a promising discovery that can lead to future developments in application of SR concepts in video compression. This gain can be computed using Bjøntegaard Delta (BD) metric [18], which calculates

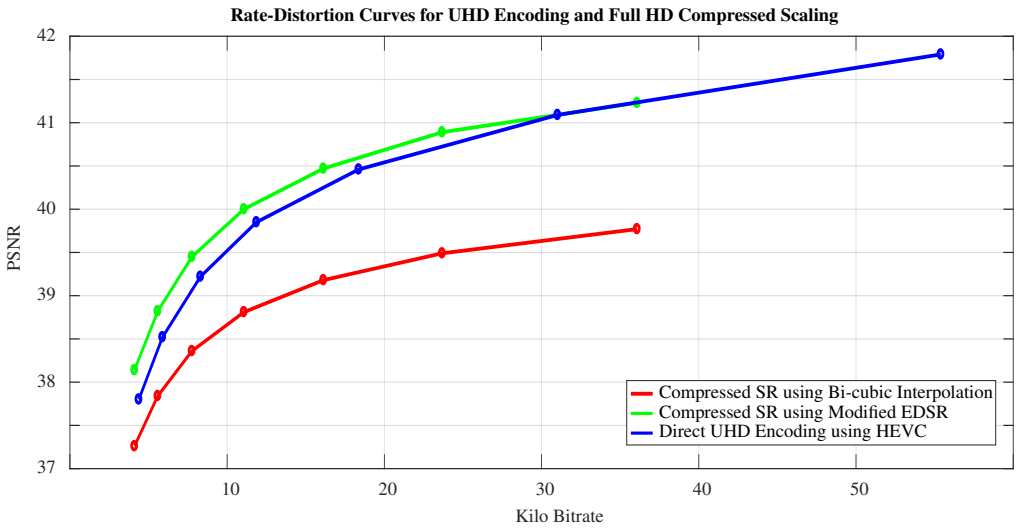


FIGURE 5.8: The rate-distortion curves for the three tested scenarios including the direct HEVC encoding of UHD content, and compressed scaling from encoded Full HD versions using modified EDSR and bi-cubic interpolation.

the area between two rate-distortion curves and quantifies the average bit-rate saving that can be achieved with the same PSNR values. The compressed SR using modified EDSR can provide a -0.33% BD-rate gain compared to the direct UHD HEVC encoding.

It is however important to note that the decoding performance is impacted significantly by the introduction of SR, in that performing high quality up-scaling at the decoder side causes major increase in the complexity of the frame reconstruction. However, the big picture in application of SR for compression seems to be bright. In addition, the QP mapping, and more importantly, application of deep learning-based SR models for high quality up-scaling of the compressed video sequences on a wide range of encoding spectrum seems to be effective and providing favorable results in terms of picture quality represented by PSNR values.

5.6 Conclusions

This chapter presented an accurate deep learning-based SR model for compressed video up-scaling to the UHD resolution. The model was based upon the powerful network architecture of the single image SR model, EDSR, and was modified and improved in terms of computational efficiency by application of lossless pooling. The proposed model was further adapted to the compressed content by devising solutions for incorporation of the compression artifacts in the training data. The presented approach was able

to be successfully applied on high resolution compressed data on the available GPU technology, providing same level of picture quality as EDSR.

In addition, the concept of QP mapping between the direct encoding of the UHD content using HEVC and compressed SR using modified EDSR was presented. It was demonstrated that high quality up-scaling of the Full HD content to the UHD resolution can lead to a similar picture quality achievable by direct UHD encoding using the appropriate QP values. The SR-based compression and QP mapping concepts were alternative ways of proving the effectiveness of the proposed modified EDSR model for application on the compressed content. Moreover, the bit-rate analysis of the compressed SR framework for QP mapping promised possibility of adapting future codecs to utilize the SR concepts.

Chapter 6

Conclusions and Future Developments

The research described in this thesis was a scientific endeavor in improvement of the Super-Resolution (SR) technology by relying on deep learning concepts and application of Convolutional Neural Networks (CNN). Section 1.2 outlined the existing challenges in designing effective SR solutions for still images and videos, and the work carried out in the context of this thesis aimed at addressing a number of those challenges based on the state-of-the-art technology described in Chapter 2, and provided several contributions to enhance the SR from different perspectives.

The contributions discussed in this thesis can be grouped into three broad categories, each presented in a separate chapter. Chapter 3 was based on a theoretical observation on training SR neural networks, within which an alternative approach was proposed for more efficient training of generative models. Chapter 4 introduced a novel pooling layer, and applied the proposed layer to the well-known SR architectures aiming at improving the inference in terms of both computation complexity and picture quality. Chapter 5 contributions were application-oriented, and focused on SR in compressed data with considerations on deploying SR technology as a compression tool for encoding high resolution content. The detailed scientific achievements of this thesis are listed in the next section.

6.1 Summary of Scientific Achievements

The first achievement was designing an efficient cost function for training SR networks. The Modified Proximity-based Cost (MPC) function was described in details in Section 3.4. The MPC function was inspired by the existing quality assessment metrics and observations on behavior of different loss functions. MPC proved to be an effective cost function that can replace the conventional Mean Squared Error (MSR) loss, and help SR networks converge much faster to the desired parameters with more than 90% decrease in the number of required back-propagations in some of the well-known architectures, as well as providing slight improvements in the quality performance of the inference.

In terms of architecture design, Section 4.2 introduced the novel lossless pooling layer, that rearranges the pixels in a tensor and provides multiple down-scaled replicas of the input signal without losing any information during the pooling operation. The lossless pooling was designed to be applicable in SR networks and provide speed-ups as well as quality enhancement during the inference.

More specifically, Section 4.3 proposed two approaches in integrating the lossless pooling layers into the existing encoder-decoder architectures for SR. The Fast Lossless Pooling Network (FLPN) aimed at reducing the inference time of the image reconstruction for high resolution content. In particular, the lossless pooling operation integrated in some of the well-known architectures led to more than 90% reduction of the computation time for creating UHD content.

Furthermore, application of the Accurate Lossless Pooling Network (ALPN) led to integration of multiple self-replicas obtained from the input image into the network, that could boost the inference performance in terms of the picture reconstruction quality by exploiting the high correlation between the input signal self-replicas. The proposed architecture provided superior picture quality compared to the baseline models in both objective and subjective assessments.

Section 5.3 proposed a deep learning-based SR model for up-scaling Full HD compressed videos to the 4K UHD resolution with considerations for model complexity and picture quality. The model was based on a powerful deep structure for still images, and it was adapted and improved using the FLPN concept to provide faster inference, which is a crucial objective for handling high resolution content. In particular, the modifications

enabled the model to process UHD content; a key achievement that was not plausible with the deep baseline architecture.

Additionally, the training stage was adapted to be able to handle the various compression artifacts that exist in the encoded content. Using HEVC and JPEG encoders, the training data was encoded with different settings, and a set of images containing different compression distortions were obtained to be used in the training of the SR model. The results proved that the model is effective in dealing with compressed material, and the quality was evaluated by comparison with bi-cubic interpolation, as well as QP mapping with direct UHD encoding.

The proposed SR model for up-scaling compressed videos triggered the discussions for application of SR as a compression solution in Section 5.4. Application of SR by coupled down-sampling and up-sampling at the encoder and decoder sides, respectively, proved to be an effective way of encoding the UHD material. The method was able to provide comparable results in terms of bit-rate and PSNR with respect to the case of direct UHD encoding using HEVC, and opened up discussions for future developments of the video encoders.

6.2 Potential Future Developments

The contributions in this thesis open new doors in enhancing the SR technology in still images and videos, and promise further developments that can be built based on the proposed methodologies. The proposed MPC function for training SR networks can be further adapted and modified to take advantage of more sophisticated feature-based terms such as incorporating the VGG loss [103] or other perceptual loss functions. Inclusion of such concepts in MPC is expected to enhance the picture quality of the trained networks in addition to providing fast convergence in SR models.

Lossless pooling layers, on the other hand, have huge potential to be applied in other non-SR domains and contribute to the performance of the deep learning architectures for various tasks. The effectiveness of the operation, along with devising more innovative ways to integrate such operation within the CNNs can be investigated further.

Another promising field that can be further studied and developed is the application of SR-based approaches in video compression. Although this thesis proved that SR-based models can provide comparable bit-rates and PSNR values to the HEVC, the integration of the SR in compression pipelines can be optimized further for providing more efficient work-flows for encoding and decoding processes. Specifically, a more stable approach for exploiting the SR concept is by weaving the process within the rate-distortion optimization stage of the encoders, and apply SR selectively on specific frames (or portions of frames) to ensure a compression gain, as well as smoother decoding process and compatibility with coding standards.

As a final note on evolution of the SR, it is fair to think the future of SR is heavily linked with how deep learning and artificial intelligence will grow in the next decade. Introduction of elaborate neural networks and complex architectures, along with integration of low level computer vision concepts in deep learning, as well as solving various existing issues in training complicated models and handling large amounts of data will eventually make ways for more robust and efficient SR solutions. Given the nature of SR, which is essentially an enhancement tool, it is not strange to see models that take SR one step further and integrate multiple enhancement chains in a single SR engine. In particular, application of higher dimensional CNNs [115] can enable performing simultaneous spatial and temporal up-scaling for videos, which is a very intriguing prospect for broadcasting and streaming applications.

Looking at the most recent developments in deep learning, with introduction of complex hierarchical structures and incorporation of various residual signals within CNN architectures, it is fair to assume that the models will continue to grow in size and complexity. When thinking about the future of deep learning, one interesting analogy comes to mind, and that is the evolution of micro-electronics and integrated circuits. Electronics started by application of simple transistors, and developed into a technology where nowadays millions of transistors can be easily accommodated into a small chip. A same trajectory can be expected in deep learning, where future architectures my include hundreds or thousands of pre-trained blocks, with each block including many convolutional layers. That would, of course, require the evolution of the hardware technology in parallel. But more importantly, it would require determination and perseverance from future researchers.

Bibliography

[1] Ultra Video Group Test Sequences. http://ultravideo.cs.tut.fi/#testsequences. Accessed: 2019-02-02.

[2] (2003). ITU-T Rec. H.264 and ISO/IEC 14496-10: Advanced Video Coding for Generic Audio-Visual Services. Technical report, ITU-T and ISO/IEC JTC 1.

[3] (2004). ISO/IEC 15948:2004 - Portable Network Graphics (PNG): Functional specification. Technical report, ISO/IEC.

[4] (2013). ITU-T Rec. H.265 and ISO/IEC 23008-2: High Efficiency Video Coding. Technical report, ITU-T and ISO/IEC JTC 1.

[5] (2015). BT.2020: Parameter Values for Ultra-High Definition Television Systems for Production and International Programme Exchange. Technical report, International Telecommunication Union.

[6] (2018). JVET-K1001: Versatile Video Coding (Draft 2). Technical report, ITU-T and ISO/IEC JTC 1.

[7] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[8] Afonso, M., Zhang, F., and Bull, D. R. (2019). Video Compression Based on Spatio-Temporal Resolution Adaptation. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):275–280.

[9] Afonso, M., Zhang, F., Katsenou, A., Agrafiotis, D., and Bull, D. (2017). Low Complexity Video Coding Based on Spatial Resolution Adaptation. In *International Conference on Image Processing*, pages 3011–3015.

[10] Agustsson, E. and Timofte, R. (2017). NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 1122–1131.

[11] Agustsson, E., Tschannen, M., Mentzer, F., Timofte, R., and Van Gool, L. (2018). Generative Adversarial Networks for Extreme Learned Image Compression. *arXiv preprint arXiv:1804.02958*.

[12] Ahn, N., Kang, B., and Sohn, K. (2018). Image Super-Resolution via Progressive Cascading Residual Network. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 904–9048.

[13] Amit, Y. and Geman, D. (1997). Shape Quantization And Recognition With Randomized Trees. *Neural Computation*, 9(7):1545–1588.

[14] Babacan, S., Molina, R., and Katsaggelos, A. (2011). Variational Bayesian Super Resolution. *IEEE Transactions on Image Processing*, 20(4):984–999.

[15] Belekos, S. P., Galatsanos, N. P., and Katsaggelos, A. K. (2010). Maximum a Posteriori Video Super-Resolution Using a New Multichannel Image Prior. *IEEE Transactions on Image Processing*, 19(6):1451–1464.

[16] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

[17] Bevilacqua, M., Roumy, A., Guillemot, C., and Alberi-Morel, M. (2012). Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding. In *British Machine Vision Conference*, pages 1–10.

[18] Bjøntegaard, G. (2008). Improvements of the BD-PSNR Model. Technical report, ITU-T SG16/Q6, 35th VCEG Meeting, Doc.VCEG-AI11.

[19] Bossen, F. (2013). Common HM Test Conditions and Software Reference Configurations. Technical report, ITU-T JCTVC-L1100.

[20] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[21] Bruna, J., Sprechmann, P., and LeCun, Y. (2016). Super-Resolution with Deep Convolutional Sufficient Statistics. In *International Conference on Learning Representations*.

[22] Burger, H. C., Schuler, C. J., and Harmeling, S. (2012). Image Denoising: Can Plain Neural Networks Compete with BM3D? In *Conference on Computer Vision and Pattern Recognition*, pages 2392–2399.

[23] Caballero, J., Ledig, C., Aitken, A. P., Acosta, A., Totz, J., Wang, Z., and Shi, W. (2017). Real-Time Video Super-Resolution with Spatio-Temporal Networks and Motion Compensation. In *Conference on Computer Vision and Pattern Recognition*, pages 2848–2857.

[24] Chang, H., Yeung, D.-Y., and Xiong, Y. (2004). Super-Resolution through Neighbor Embedding. In *Conference on Computer Vision and Pattern Recognition*, pages 275–282.

[25] Chen, Y., Xie, Y., Zhou, Z., Shi, F., Christodoulou, A. G., and Li, D. (2018). Brain MRI Super Resolution using 3D Deep Densely Connected Neural Networks. In *International Symposium on Biomedical Imaging*, pages 739–742.

[26] Criminisi, A., Shotton, J., Criminisi, A., and Shotton, J. (2013). *Decision Forests for Computer Vision and Medical Image Analysis*. Springer.

[27] Cui, Z., Chang, H., Shan, S., Zhong, B., and Chen, X. (2014). Deep Network Cascade for Image Super-resolution. In *European Conference on Computer Vision*, pages 49–64.

[28] Dai, Q., Yoo, S., Kappeler, A., and Katsaggelos, A. K. (2015). Dictionary-Based Multiple Frame Video Super-Resolution. In *International Conference on Image Processing*, pages 83–87.

[29] Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[30] Dong, C., Loy, C. C., He, K., and Tang, X. (2014). Learning a Deep Convolutional Network for Image Super-Resolution. In *European Conference on Computer Vision*, pages 184–199.

[31] Dong, C., Loy, C. C., He, K., and Tang, X. (2016a). Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307.

[32] Dong, C., Loy, C. C., and Tang, X. (2016b). Accelerating the Super-Resolution Convolutional Neural Network. In *European Conference on Computer Vision*, pages 184–199.

[33] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12(7):2121–2159.

[34] Farsiu, S., Robinson, M. D., Elad, M., and Milanfar, P. (2004). Fast and Robust Multiframe Super Resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344.

[35] Freedman, G. and Fattal, R. (2011). Image and Video Upscaling from Local Self-examples. *ACM Transactions on Graphics*, 30(2):12:1–12:11.

[36] Freeman, W. T., Jones, T. R., and Pasztor, E. C. (2002). Example-Based Super-Resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65.

[37] Fu, C.-M., Alshina, E., Alshin, A., Huang, Y.-W., Chen, C.-Y., Tsai, C.-Y., Hsu, C.-W., Lei, S., Park, J.-H., and Han, W. (2012). Sample Adaptive Offset in the HEVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1755–1764.

[38] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). Texture Synthesis Using Convolutional Neural Networks. In *Conference on Neural Information Processing Systems*, pages 262–270.

[39] Georgis, G., Lentaris, G., and Reisis, D. (2016). Reduced Complexity Superresolution for Low-Bitrate Video Compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(2):332–345.

[40] Glasner, D., Bagon, S., and Irani, M. (2009). Super-Resolution from a Single Image. In *International Conference on Computer Vision*, pages 349–356.

[41] Gohshi, S. (2015). Real-Time Super Resolution Algorithm for Security Cameras. In *International Joint Conference on e-Business and Telecommunications*, pages 92–97.

[42] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[43] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In *Conference on Neural Information Processing Systems*, pages 2672–2680.

[44] Hardie, R. C., Barnard, K. J., and Armstrong, E. E. (1997). Joint MAP Registration and High-Resolution Image Estimation using a Sequence for Undersampled Images. *IEEE Transactions on Image Processing*, 6(12):1621–1633.

[45] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. pages 1026–1034.

[46] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[47] Hennings-Yeomans, P. H., Kumar, B. V. K. V., and Baker, S. (2009). Robust Low-Resolution Face Identification and Verification using High-Resolution Features. In *International Conference on Image Processing*, pages 33–36.

[48] Huang, J.-B., Singh, A., and Ahuja, N. (2015a). Single Image Super-Resolution from Transformed Self-Exemplars. In *Conference on Computer Vision and Pattern Recognition*, pages 5197–5206.

[49] Huang, Y., Wang, W., and Wang, L. (2015b). Bidirectional Recurrent Convolutional Networks for Multi-Frame Super-Resolution. In *Conference on Neural Information Processing Systems*, pages 235–243.

[50] Huang, Y., Wang, W., and Wang, L. (2018). Video super-resolution via bidirectional recurrent convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):1015–1028.

[51] Hudson, G., Lger, A., Niss, B., and Sebestyn, I. (2017). JPEG at 25: Still Going Strong. *IEEE MultiMedia*, 24(2):96–103.

[52] Hughes, C. G. and Ramsey, M. S. (2010). Super-Resolution of THEMIS Thermal Infrared Data: Compositional Relationships of Surface Units Below the 100 Meter Scale on Mars. *Icarus*, 208(2):704 – 720.

[53] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 1647–1655.

[54] Jiang, K., Wang, Z., Yi, P., and Jiang, J. (2018). A Progressively Enhanced Network for Video Satellite Imagery Superresolution. *IEEE Signal Processing Letters*, 25(11):1630–1634.

[55] Jo, Y., Oh, S. W., Kang, J., and Kim, S. J. (2018). Deep Video Super-Resolution Network Using Dynamic Upsampling Filters Without Explicit Motion Compensation. In *Conference on Computer Vision and Pattern Recognition*, pages 3224–3232.

[56] Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *European Conference on Computer Vision*, pages 694–711.

[57] Jolicoeur-Martineau, A. (2018). The Relativistic Discriminator: A Key Element Missing from Standard GAN. *arXiv preprint arXiv:1807.00734*.

[58] Kamimura, K., Tsumura, N., Nakaguchi, T., Miyake, Y., and Motomura, H. (2007). Video Super-Resolution using Texton Substitution. In *ACM SIGGRAPH Posters*, page 63.

[59] Kanemura, A., Maeda, S.-i., and Ishii, S. (2009). Superresolution with Compound Markov Random Fields via the Variational EM Algorithm. *Elsevier Journal on Neural Networks*, 22(7):1025–1034.

[60] Kappeler, A., Yoo, S., Dai, Q., and Katsaggelos, A. K. (2016a). Super-Resolution of Compressed Videos using Convolutional Neural Networks. In *International Conference on Image Processing*, pages 1150–1154.

[61] Kappeler, A., Yoo, S., Dai, Q., and Katsaggelos, A. K. (2016b). Video Super-Resolution With Convolutional Neural Networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122.

[62] Kato, T., Hino, H., and Murata, N. (2015). Multi-Frame Image Super Resolution Based on Sparse Coding. *Elsevier Journal on Neural Networks*, 66(3):64–78.

[63] Kim, J., Kwon Lee, J., and Mu Lee, K. (2016a). Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 1646–1654.

[64] Kim, J., Kwon Lee, J., and Mu Lee, K. (2016b). Deeply-Recursive Convolutional Network for Image Super-Resolution. In *Conference on Computer Vision and Pattern Recognition*, pages 1637–1645.

[65] Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[66] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Conference on Neural Information Processing Systems*, pages 1097–1105.

[67] Kumar, V. R. V., Vidya, A., Sharumathy, M., and Kanizohi, R. (2017). Super Resolution Enhancement of Medical Image using Quaternion Wavelet Transform with SVD. In *International Conference on Signal Processing, Communication and Networking*, pages 1–7.

[68] Lai, W., Huang, J., Ahuja, N., and Yang, M. (2018). Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

[69] Lai, W.-S., Huang, J.-B., Ahuja, N., and Yang, M.-H. (2017). Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution. In *Conference on Computer Vision and Pattern Recognition*, pages 5835–5843.

[70] Laude, T. and Ostermann, J. (2016). Deep Learning-Based Intra Prediction Mode Decision for HEVC. In *Picture Coding Symposium*, pages 1–5.

[71] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551.

[72] Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *Conference on Computer Vision and Pattern Recognition*, pages 4681–4690.

[73] Lewis, D. D. (1998). Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In *European Conference on Machine Learning*, pages 4–15.

[74] Li, L., Yu, Q., Yuan, Y., Shang, Y., Lu, H., and Sun, X. (2009). Super-Resolution Reconstruction and Higher-Degree Function Deformation Model Based Matching for Chang'E-1 Lunar Images. *Science in China Series E: Technological Sciences*, 52(12):3468.

[75] Li, T., Xu, M., and Deng, X. (2017). A Deep Convolutional Neural Network Approach for Complexity Reduction on Intra-Mode HEVC. In *International Conference on Multimedia and Expo*, pages 1255–1260.

[76] Liao, R., Tao, X., Li, R., Ma, Z., and Jia, J. (2015). Video Super-Resolution via Deep Draft-Ensemble Learning. In *International Conference on Computer Vision*, pages 531–539.

[77] Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M. (2017). Enhanced Deep Residual Networks for Single Image Super-Resolution. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 1132–1140.

[78] Liu, C. and Sun, D. (2014). On Bayesian Adaptive Video Super Resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):346–360.

[79] Liu, D., Wang, Z., Fan, Y., Liu, X., Wang, Z., Chang, S., and Huang, T. (2017). Robust Video Super-Resolution with Learned Temporal Dynamics. In *International Conference on Computer Vision*, pages 2526–2534.

[80] Liu, D., Wang, Z., Nasrabadi, N., and Huang, T. (2016a). Learning a Mixture of Deep Networks for Single Image Super-Resolution. In *Asian Conference on Computer Vision*, pages 145–156.

[81] Liu, D., Wang, Z., Wen, B., Yang, J., Han, W., and Huang, T. S. (2016b). Robust Single Image Super-Resolution via Deep Networks With Sparse Prior. *IEEE Transactions on Image Processing*, 25(7):3194–3207.

[82] Liu, Z. and Cui, C. (2018). A New Low Bit-Rate Coding Scheme for Ultra High Definition Video Based on Super-Resolution Reconstruction. In *Conference on Computer and Communication Engineering Technology*, pages 325–329.

[83] Ma, Z., Liao, R., Tao, X., Xu, L., Jia, J., and Wu, E. (2015). Handling Motion Blur in Multi-Frame Super-Resolution. In *Conference on Computer Vision and Pattern Recognition*, pages 5224–5232.

[84] Makansi, O., Ilg, E., and Brox, T. (2017). End-to-End Learning of Video Super-Resolution with Motion Compensation. In *German Conference on Pattern Recognition*, pages 203–214.

[85] Mao, X., Shen, C., and Yang, Y. (2016). Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections. In *Neural Information Processing Systems*, pages 2802–2810.

[86] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *International Conference on Computer Vision*, pages 416–423.

[87] Nagi, J., Ducatelle, F., Caro, G. A. D., Cirean, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J., and Gambardella, L. M. (2011). Max-Pooling Convolutional Neural Networks for Vision-Based Hand Gesture Recognition. In *International Conference on Signal and Image Processing Applications*, pages 342–347.

[88] Nair, V. and E. Hinton, G. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *International Conference on Machine Learning*, pages 807–814.

[89] Nemoto, H., Hanhart, P., Korshunov, P., and Ebrahimi, T. (2014). Ultra-Eye: UHD and HD images eye tracking dataset. In *Workshop on Quality of Multimedia Experience (QoMEX)*, pages 39–40.

[90] Norkin, A., Bjøntegaard, G., Fuldseth, A., Narroschke, M., Ikeda, M., Andersson, K., Zhou, M., and Auwera, G. V. D. (2012). HEVC Deblocking Filter. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1746–1754.

[91] Olshausen, B. and Field, D. (1996). Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images. *Nature*, 381(6):607–609.

[92] Osendorfer, C., Soyer, H., and van der Smagt, P. (2014). Image Super-Resolution with Fast Approximate Convolutional Sparse Coding. In *Conference on Neural Information Processing*, pages 250–257.

[93] Sajjadi, M. S. M., Schölkopf, B., and Hirsch, M. (2017). EnhanceNet: Single Image Super-Resolution through Automated Texture Synthesis. In *International Conference on Computer Vision*, pages 4501–4510.

[94] Sajjadi, M. S. M., Vemulapalli, R., and Brown, M. (2018). Frame-Recurrent Video Super-Resolution. In *Conference on Computer Vision and Pattern Recognition*, pages 6626–6634.

[95] Salvador, J. and Prez-Pellitero, E. (2015). Naive Bayes Super-Resolution Forest. In *International Conference on Computer Vision*, pages 325–333.

[96] Santamaria, M., Izquierdo, E., Blasi, S., and Mrak, M. (2018). Estimation of Rate Control Parameters for Video Coding Using CNN. In *Visual Communications and Image Processing*, pages 1–4.

[97] Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *International Conference on Artificial Neural Networks: Part III*, pages 92–101.

[98] Schuler, C. J., Burger, H. C., Harmeling, S., and Schlkopf, B. (2013). A Machine Learning Approach for Non-blind Image Deconvolution. In *Conference on Computer Vision and Pattern Recognition*, pages 1067–1074.

[99] Schulter, S., Leistner, C., and Bischof, H. (2015). Fast and Accurate Image Upscaling with Super-Resolution Forests. In *Conference on Computer Vision and Pattern Recognition*, pages 3791–3799.

[100] Shelhamer, E., Long, J., and Darrell, T. (2017). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651.

[101] Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016). Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *Conference on Computer Vision and Pattern Recognition*, pages 1874–1883.

[102] Shocher, A., Cohen, N., and Irani, M. (2018). "Zero-Shot" Super-Resolution using Deep Internal Learning. In *Conference on Computer Vision and Pattern Recognition*, pages 3118–3126.

[103] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[104] Song, B. C., Jeong, S., and Choi, Y. (2011). Video Super-Resolution Algorithm Using Bi-Directional Overlapped Block Motion Compensation and On-the-Fly Dictionary Training. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(3):274–285.

[105] Song, R., Liu, D., Li, H., and Wu, F. (2017). Neural Network-Based Arithmetic Coding of Intra Prediction Modes in HEVC. In *Visual Communications and Image Processing*, pages 1–4.

[106] Sullivan, G. J., Ohm, J.-R., Han, W., and Wiegand, T. (2012). Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668.

[107] Sun, J., Zheng, N., Tao, H., and Shum, H.-Y. (2003). Image Hallucination with Primal Sketch Priors. In *Conference on Computer Vision and Pattern Recognition*, pages 729–736.

[108] Tai, Y., Yang, J., and Liu, X. (2017). Image Super-Resolution via Deep Recursive Residual Network. In *Conference on Computer Vision and Pattern Recognition*, pages 2790–2798.

[109] Tao, X., Gao, H., Liao, R., Wang, J., and Jia, J. (2017). Detail-Revealing Deep Video Super-Resolution. In *International Conference on Computer Vision*, pages 4482–4490.

[110] Tieleman, T. and Hinton, G. (2012). Lecture 6.5 - RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[111] Timofte, R., Smet, V. D., and Gool, L. V. (2013). Anchored Neighborhood Regression for Fast Example-Based Super-Resolution. In *International Conference on Computer Vision*, pages 1920–1927.

[112] Timofte, R., Smet, V. D., and Gool, L. V. (2014). A+: Adjusted Anchored Neighborhood Regression for Fast Super-Resolution. In *Asian Conference on Computer Vision*, pages 111–126.

[113] Tong, T., Li, G., Liu, X., and Gao, Q. (2017). Image Super-Resolution Using Dense Skip Connections. *International Conference on Computer Vision*, pages 4809–4817.

[114] Toutounchi, F., Guerra Ones, V., and Izquierdo, E. (2017). An Efficient Super-Resolution Approach based on Sparse Representation. In *International Workshop on Multimedia Signal Processing*, pages 1–6.

[115] Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning Spatiotemporal Features with 3D Convolutional Networks. In *International Conference on Computer Vision*, pages 4489–4497.

[116] Union, I. T. HM Reference Software. `https://hevc.hhi.fraunhofer.de/HM-doc/`. Accessed: 2019-02-02.

[117] Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. (2018a). High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *Conference on Computer Vision and Pattern Recognition*, pages 8798–8807.

[118] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., and Loy, C. C. (2018b). ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. In *European Conference on Computer Vision Workshops*, pages 63–79.

[119] Wang, Y., Perazzi, F., McWilliams, B., Sorkine-Hornung, A., Sorkine-Hornung, O., and Schroers, C. (2018c). A Fully Progressive Approach to Single-Image Super-Resolution. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 977–97709.

[120] Wang, Z., Liu, D., Yang, J., Han, W., and Huang, T. (2015a). Deep Networks for Image Super-Resolution with Sparse Prior. In *International Conference on Computer Vision*, pages 370–378.

[121] Wang, Z., Yang, Y., Wang, Z., Chang, S., Han, W., Yang, J., and Huang, T. S. (2015b). Self-Tuned Deep Super Resolution. In *Computer Vision and Pattern Recognition Workshops*, pages 1–8.

[122] Weerakkody, R., Naccari, M., and Mrak, M. (2013). UHD Test Sequences. Technical report, JCTVC-O0332.

[123] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. (2003). Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576.

[124] Xu, B., Pan, X., Zhou, Y., Li, Y., Yang, D., and Chen, Z. (2017). CNN-Based Rate-Distortion Modeling for H.265/HEVC. In *Visual Communications and Image Processing*, pages 1–4.

[125] Yandex, A. B. and Lempitsky, V. (2015). Aggregating Local Deep Features for Image Retrieval. In *International Conference on Computer Vision*, pages 1269–1277.

[126] Yang, C.-Y. and Yang, M.-H. (2013). Fast Direct Super-Resolution by Simple Functions. *International Conference on Computer Vision*, pages 561–568.

[127] Yang, J., Lin, Z., and Cohen, S. (2013). Fast Image Super-Resolution Based on In-Place Example Regression. In *Conference on Computer Vision and Pattern Recognition*, pages 1059–1066.

[128] Yang, J., Wang, Z., Lin, Z., Cohen, S., and Huang, T. (2012). Coupled Dictionary Training for Image Super-Resolution. *IEEE Transactions on Image Processing*, 21(8):3467–3478.

[129] Yang, J., Wright, J., Huang, T., and Ma, Y. (2008). Image Super-Resolution as Sparse Representation of Raw Image Patches. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[130] Yang, J., Wright, J., Huang, T. S., and Ma, Y. (2010). Image Super-Resolution via Sparse Representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873.

[131] Yuan, Y., Liu, S., Zhang, J., Zhang, Y., Dong, C., and Lin, L. (2018). Unsupervised Image Super-Resolution Using Cycle-in-Cycle Generative Adversarial Networks. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 814–81409.

[132] Zamani, N. A., Darus, M. Z. A., Abdullah, S. N. H. S., and Nordin, M. J. (2011). Multiple-Frames Super-Resolution for Closed Circuit Television Forensics. In *International Conference on Pattern Analysis and Intelligence Robotics*, pages 36–40.

[133] Zeiler, M. D. and Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision*, pages 818–833.

[134] Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 2528–2535.

[135] Zeiler, M. D., Taylor, G. W., and Fergus, R. (2011). Adaptive Deconvolutional Networks for Mid and High Level Feature Learning. In *International Conference on Computer Vision*, pages 2018–2025.

[136] Zeyde, R., Elad, M., and Protter, M. (2012). On Single Image Scale-up Using Sparse-representations. In *International Conference on Curves and Surfaces*, pages 711–730.

[137] Zhang, H. (2004). The Optimality of Naive Bayes. In *International Florida Artificial Intelligence Research Society Conference*, pages 562–567.

[138] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. (2017). StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks. In *International Conference on Computer Vision*, pages 5908–5916.

[139] Zhang, Y., Tian, Y., Kong, Y., Zhong, B., and Fu, Y. (2018). Residual Dense Network for Image Super-Resolution. In *Conference on Computer Vision and Pattern Recognition*, pages 2472–2481.

[140] Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2017). Loss Functions for Image Restoration With Neural Networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57.

[141] Zhao, N., Wei, Q., Basarab, A., Kouam, D., and Tourneret, J. (2016). Single Image Super-Resolution of Medical Ultrasound Images using a Fast Algorithm. In *International Symposium on Biomedical Imaging*, pages 473–476.

[142] Zhi-Song, L. and Siu, W. (2018). Cascaded Random Forests for Fast Image Super-Resolution. In *International Conference on Image Processing*, pages 2531–2535.